TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Archivace posloupností hodnot měřených veličin elektrické energie

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Martin Vondráček**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Archive for time series values measured from power monitoring

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information Technology

Author: **Martin Vondráček**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Archivace posloupností hodnot měřených veličin elektrické energie

Jméno a příjmení: **Martin Vondráček**
Osobní číslo: M16000063
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Seznamte se s obsahem archivů měření kvality elektrické energie a s významem jednotlivých ukládaných veličin.
2. Na větším množství různých měření sledujte charakteristické vlastnosti posloupností vybraných měřených hodnot a na základě pozorování navrhněte vhodný způsob kódování a komprimace těchto dat.
3. Důkladně prověřte výkonnost Vámi navržených metod a použitých kompresních knihoven v prostředí .NET(C#) po stránce objemu dat a rychlosti čtení i zápisu.
4. Dosažené výsledky shrňte a diskutujte přehledným způsobem v průvodní zprávě.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	tištěná/elektronická

Seznam odborné literatury:

[1] ALBU, Mihaela M., et al. Monitoring voltage and frequency in smart distribution grids. A case study on data compression and accessibility. In: IEEE PES General Meeting. IEEE, 2010. p. 1-6.

[2] RINGWELSKI, Martin, et al. The Hitchhiker's guide to choosing the compression algorithm for your smart meter data. In: Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International. IEEE, 2012. p. 935-940.

[3] TCHEOU, Michel P., et al. The compression of electric signal waveforms for smart grids: State of the art and future trends. IEEE Transactions on Smart Grid, 2014, 5.1: 291-302.

[4] NELSON, Mark; GAILLY, Jean-Loup. The data compression book. New York: M&t Books, 1996.

[5] MSDN. System.IO.Compression Namespace [online]. [cit. 2016-10-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/system.io.compression(v=vs.110).aspx.

Vedoucí práce:	Ing. Jan Kraus, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	10. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci 10. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

28. 4. 2019 Martin Vondráček

Poděkování

Děkuji svému vedoucímu Janu Krausovi za pomoc a čas, který mi věnoval při vytváření této práce. Další dík patří Janu Vyšohlídovi za technickou podporu při vytváření této zprávy. Nakonec chci poděkovat svým rodičům za hojnou psychickou i materiální podporu.

Abstrakt

Tato práce se věnuje různým způsobům komprese binárních souborů, které obsahují měření elektrických veličin. Po představení používaných kompresních algoritmů a kódování dat je čtenář seznámen s ukládanými veličinami vyskytujícími se v archivech měření elektrické energie. Následuje testování zvolených kompresních algoritmů na reálných datech. Ve všech prováděných testech se ke kompresi používají bezeztrátové algoritmy Lzma, Bzip2 a Deflate, u kterých se analyzuje a zaznamenává jejich efektivita a časová náročnost. Před kompresí těmito algoritmy se data různě předběžně zpracovávají, aby bylo možné porovnat efektivitu komprese při různé reprezentaci dat v souborech. Výsledky jsou mezi sebou porovnány a následně se vyvozují závěry o vlastnostech a chování jednotlivých způsobů komprese. Za nejvýhodnější způsob předběžného zpracování je označena kombinace delta kódování a pevné řádové čárky.

Druhá část práce se zabývá představením zatím netestovaného ztrátového způsobu komprese. Při předběžném zpracování se využívá třída přesnosti měřících přístrojů k výhodné úpravě dat. Navržená technika je poté analyzována z hlediska svých parametrů a ozkoušena na datech. Testováno je i chování algoritmu při různé reprezentaci dat. Opět jsou ozkoušeny kompresní algoritmy Lzma, Bzip2 a Deflate. Podrobné výsledky jsou obsaženy v této zprávě. Jako nejvýkonnější kompresní algoritmus je zvolen Lzma.

Všechny kompresní techniky jsou testované na větším množství dat. (Počet hodnot pro jedno měření elektrické veličiny se pohybuje mezi 4 a 10 miliony hodnot.) Veškeré testování se provádí v C#.

Klíčová slova: ztrátová a bezeztrátová komprese, kompresní algoritmy, třída přesnosti, měření elektrické energie, Lzma, Bzip2, Deflate

Abstract

This work deals with various compression techniques, that are used to compress binary files, that contain measurements of electricity. Following the presentation of some popular compression algorithms and encodings, reader is acquainted with measurements of electrical energy stored in electrical energy archives. Chosen compression techniques are tested on real data. During all tests, tested algorithms are Bzip2, Lzma and Deflate. Their performace is rated by their compression ratio and time duration of the compression. Before the compression by the mentioned algorithms, the data is variously pre-processed so that I can evaluate different data representations. Results are evaluated and algorithm's important traits are analysed. Combination of delta encoding and fixed point representation is chosen as the best suited compression technique.

Second part of the work is about introduction of new compression process. This technique uses accuracy class in the pre-processing part of compression to homogenize the data. Results of various tests of this technique as well as it's detailed description is recorded here. Lzma is chosen as the most suitable compression algorithm.

All the compression techniques are tested on the large amount of data (One measurement contains form 4 to 10 milion values.) The tests are conducted in C#.

Keywords: lossy and lossless compression, compression algorithms, accuracy class, electricity measurement, Lzma, Bzip2, Deflate

Obsah

Seznam ilustrací

Seznam tabulek

Seznam zkratek

CR	Stupeň komprese
TP	Třída přesnosti
PT	Pásmo tolerance / Povolené pásmo
RMSE	Střední kvadratická chyba
MSE	Střední absolutní chyba
U	Napětí
I	Proud
f	Frekvence
P	Činný výkon
Q	Jalový výkon
THD	Celkové harmonické zkreslení

1 Úvod

Potřeba uchovávání stále většího množství měřených charakteristik elektrické energie existuje již dlouho. Jedním z hlavních důvodů je rozvoj inteligentních sítí (anglicky smart grid), které jsou automatizované a umí regulovat elektrické veličiny dle potřeby odběratele. Kvůli analýze a monitorování el. energie se nepřetržitě ukládá a monitoruje velké množství měření a proto je třeba ukládat tato data v co nejkompaktnější formě. Kromě nároků na úložiště může být při výběru správné komprese důležitý i nárok na operační paměť, rychlost komprese či dekomprese a obecně rychlost čtení a zápisu dat.

V této práci věnující se efektivnímu ukládání měření elektrických veličin se budu zabývat analýzou ztrátových i bezeztrátových způsobů kódování veličin, využití některých známých kompresních algoritmů a jejich formátů (především 7z, bz2 a zip) k uložení souborů s naměřenými hodnotami v co nejkompaktnější formě.

Práce bude začínat rešerší potřebných znalostí a některých zjištění z jiných prací, které se podobnou tématikou zabývají, analýzou různých ztrátových a bezeztrátových kompresních algoritmů a kódování. Následně popíše měření elektrických veličin, se kterými budu pracovat. Z nabytých poznatků se pokusím navrhnout co nejvhodnější způsoby archivace těchto dat, které také otestuji. Úspěšnost testovaných technik zde bude analyzována, a to především z hlediska úspěšnosti komprese a časových nároků na kompresi. U kompresních formátů a kódování se budu snažit najít klady a zápory v jejich použití. Veličiny jsou původně uloženy v CEA souborech obsahujících velká množství měřených hodnot. Tyto údaje budou ztrátově i bezeztrátově předběžně zakódovány, následně exportovány do binárních souborů a nakonec zabaleny několika komprimačními algoritmy .

Výsledkem práce nemá být program či aplikace a číselné výsledky v oblasti efektivity a doby komprese se nemají považovat za pevně stanovené chování, protože vlastnosti komprese jsou proměnlivé v závislosti na komprimovaných datech a na výkonnosti používaného zařízení. Výsledkem by mělo být především poměrové porovnání mezi algoritmy a kódováními. Pokud se bude prezentovat nový způsob komprese, bude nejdůležitější přesně zaznamenat jeho princip a vlastnosti. Důraz by neměl být kladen pouze na konkrétní čísla u konkrétních veličin, ale na všestrannost algoritmů a jejich použitelnost na všechny druhy dat, která bude třeba archivovat. Výsledky se co nejpřehledněji zaznamenají v této zprávě.

2 Teoretická část

Nejdůležitější způsob, jakým lze kompresi dat rozdělit, je dělení na ztrátovou a bezeztrátovou kompresi. Při bezeztrátové kompresi nesmí transformace ukládaných dat vyústit ve ztrátu jakékoliv informace. Cenou za uchování celé informace je ale menší efektivita komprese. Při ztrátové kompresi se část dat "zahazuje" výměnou za lepší stupeň komprese. Při ztrátové kompresi se tedy rozhodujeme, jaké množství informací jsme ochotni obětovat pro lepší úroveň komprese. Častý způsob využití obou druhů komprese je homogenizace dat ztrátovými kompresními algoritmy, která umožní efektivnější kompresi některým bezeztrátovým algoritmem. (Uspořádaná a stejnorodá data se většinou komprimují lépe.) Před samotnými testy je třeba představit a popsat některé pojmy a algoritmy, jelikož je budu používat jak k testování, tak k následné analýze a popisu testů v této zprávě.

2.1 Důležité pojmy

2.1.1 Stupeň komprese

V oboru komprese dat často využívaný pojem (anglicky compression ratio), zkráceně **CR**. Tato veličina neobsahuje informaci o velikosti původního souboru nebo souboru po kompresi. Vyjadřuje pouze efektivitu komprese vyčíslením podílu velikosti souboru před a po kompresi. (Tudíž platí, že čím větší CR, tím lepší komprese.) Jedná se o bezrozměrnou veličinu.

$$CR = \frac{velikost\ původního\ souboru}{velikost\ zkomprimovaného\ souboru}$$

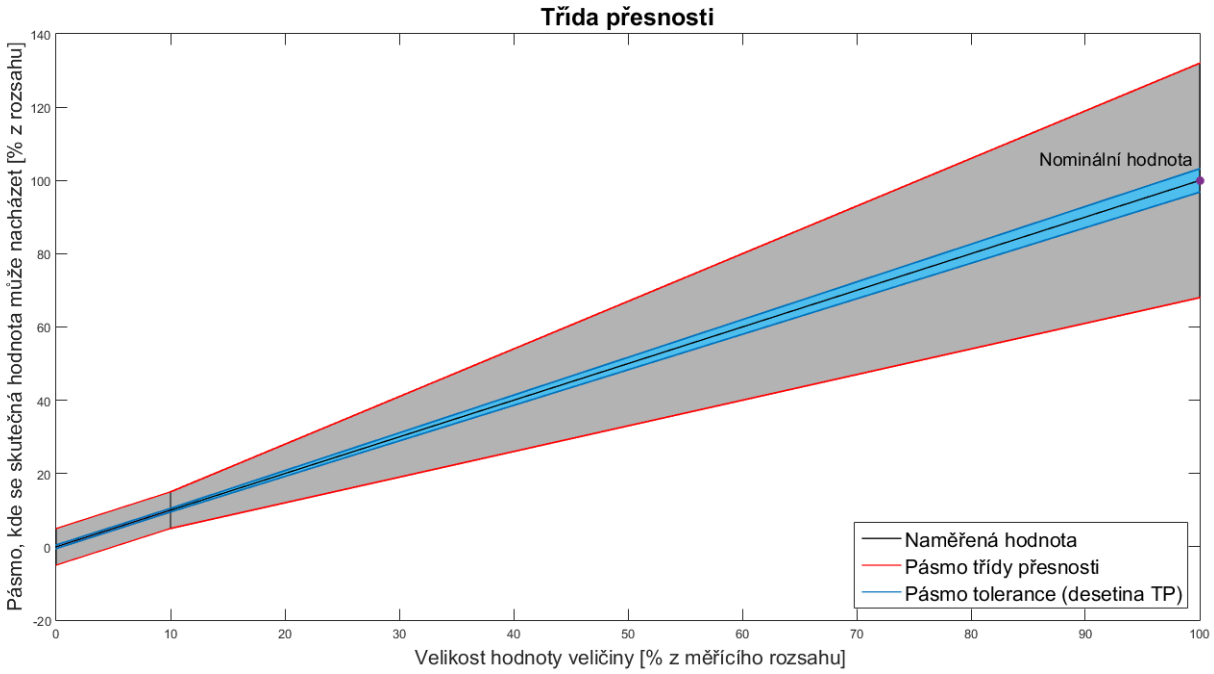
Někdy se vyjadřuje procentuálně jako podíl souboru po kompresi a souboru před kompresí vynásobený stem. V této práci budu pracovat s první variantou.

2.1.2 Třída přesnosti

Při měření elektrických veličin se nelze vyhnout nepřesnostem měření, takže naměřená hodnota neodpovídá skutečné hodnotě veličiny. Měřící přístroje vždy pracují s tzv. třídou přesnosti, která pro dané měřící rozsahy udává možnou procentuální odchylku od naměřené hodnoty. Odchylka definuje, v jakém pásmu od naměřené hodnoty se skutečná hodnota veličiny může pohybovat. Na ilustračním obrázku 2.1 je znázorněno, jak by vypadal průběh třídy přesnosti 20. S větší naměřenou hodnotou se zvětšuje i možná numerická odchylka, přestože procentuálně je třída přesnosti

stejná. (Viz rozdíl mezi velikostí pásma TP při naměřených hodnotách o velikosti 40 a 80% z měřícího rozsahu.) Čím větší třída přesnosti, tím širší jsou povolená pásma. Při změně velikosti hodnoty není doporučeno využívat celé pásmo třídy přesnosti a raději se pro práci vymezí pouze určitá část TP, tzv. pásmo tolerance (Viz obr. 2.1, kde je použito povolené pásmo o velikosti 10% z pásma třídy přesnosti.)

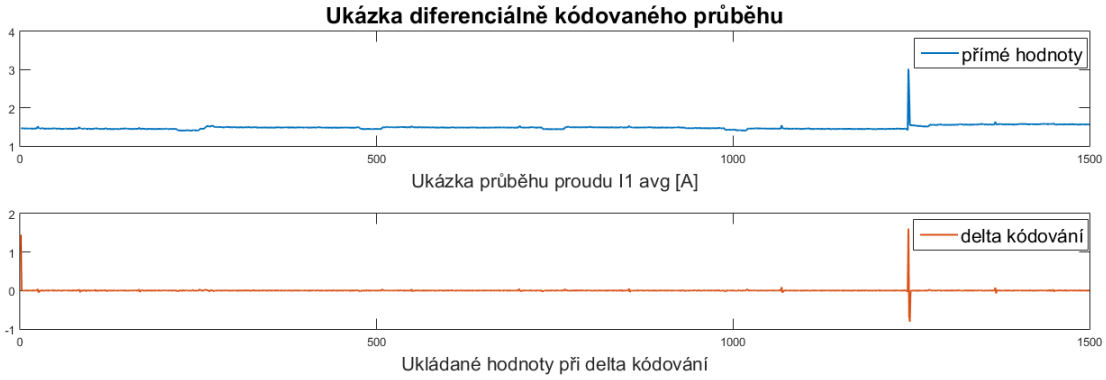
Třídu přesnosti můžeme využít k předběžnému ztrátovému zpracování dat, které zlepší výkon některých kompresních algoritmů. Jelikož se hodnota může pohybovat kdekoliv uvnitř pásma tolerance, můžeme k uložení volit hodnoty co nejvhodnější ke komprimaci nebo např. zmenšit počet ukládaných desetinných míst a podobně.



Obrázek 2.1: Graf třídy přesnosti. Měřící rozsah je určen nominální hodnotou, nominální hodnota tedy odpovídá 100% z měřícího rozsahu. Při měření hodnot blížících se nule mají měřící přístroje samozřejmě problém dodržet TP, protože je dovolená odchylka stále menší. Proto se na začátku pásma třídy přesnosti nachází oblast, která se neřídí dle TP, ale podle lineární odchylky.

2.1.3 Diferenciální kódování

Při diferenciálním, často nazývaném také jako **delta** kódování neukládáme naměřenou hodnotu přímo, nýbrž ukládáme pouze její rozdíl od hodnoty předchozí. Ve výsledku se bude ukládat číslo značně menší, což zlepší následnou kompresi do libovolného formátu. Zároveň žádnou informaci neztrácíme, metoda se tedy dá označit za bezeztrátovou.



Obrázek 2.2: Rozdíl mezi normálním a diferenciálně kódovaným průběhem

Nevýhodou této techniky je, že dochází ke střídání kladných a záporných čísel, což může mít negativní dopad na stupeň komprese. Další nevýhodou je, že přímo můžeme číst pouze změny ve velikosti veličiny. Pokud chceme znát přesnou hodnotu, musíme číst hodnoty od začátku měření a sčítat je. Proto považuji z hlediska rychlosti čtení přímé hodnoty za výhodné ukládat diferenciálně zakódovaná měření v souborech co nejmenší přijatelné velikosti.

2.1.4 Pevné a plovoucí řádové čárky

Reprezentace čísel pomocí pevné řádové čárky (anglicky **fixed point**) a plovoucí řádové čárky (angl. floating point) je způsob ukládání reálných čísel jako datový typ Integer, ve kterém je určitý počet cifer určen pro uložení čísel za desetinnou čárkou. U pevné řádové čárky je počet desetinných míst pevně dán, zatímco u plovoucí řádové čárky se počet desetinných míst mění a tudíž se musí ukládat spolu s číslem i informace o počtu desetinných míst.

Způsob ukládání čísel pomocí pevné řádové čárky ilustruji na příkladu: při ukládání hodnoty 235,893 jako Integer s pevnou řádovou čárkou na tři desetinná místa ukládám číslo jako 235 893. Pokud se spokojím s uložením pouze dvou desetinných míst, číslo uložím jako 23 589. Zde dochází ke ztrátové kompresi, ale ukládané číslo se zmenší.

Při tomto způsobu kódování je důležité vědět, kolik desetinných míst je třeba uchovávat a jakých maximálních hodnot budou zakódovaná čísla nabývat. Podle zmíněných informací lze vybrat nejvhodnější datový typ, např.: Pokud máme jistotu, že nebude třeba uchovávat čísla větší než 32 762, můžeme místo Int32 (4B) použít Int16/ Short (2B). Soubor by tak měl ještě před aplikací některého z kompresních algoritmů poloviční velikost. V praxi je ale tato hodnota většinou nedostatečná a proto bude vhodnější zvolit Int32. Podrobněji se této problematice věnuje ve své práci Pavel Tišnovský [1].

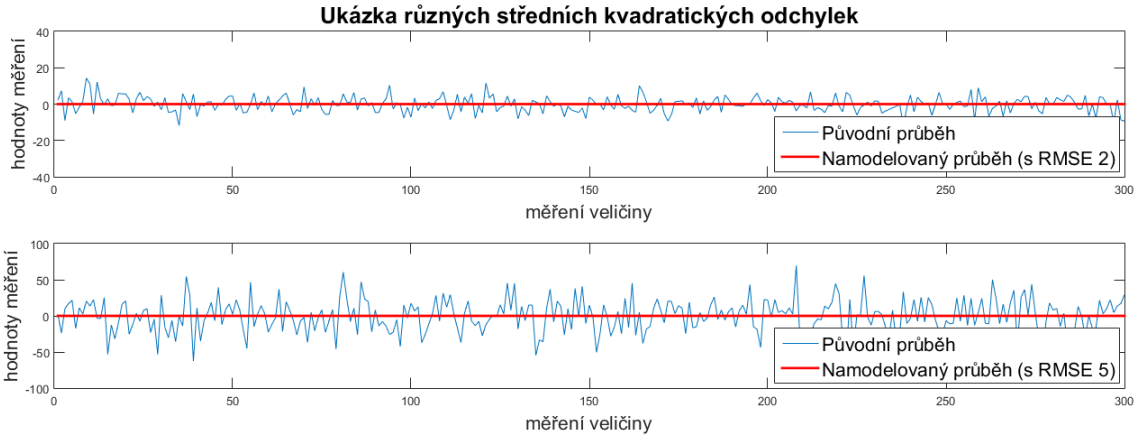
2.1.5 Střední kvadratická chyba (RMSE) a Střední absolutní odchylka (MSE)

RMSE je ukazatel rozdílu mezi naměřenými hodnotami a mezi ukládaným modelem. V teorii pravděpodobnosti se pro RMSE používá označení rozptyl a vyjadřuje odchylku od střední hodnoty. V této práci ji využijeme při ztrátové kompresi, kde se jednotlivé deviace hodnot z namodelovaného průběhu a hodnot z původního modelu umocní na druhou, z mocnin se vyjádří průměr a následně zase odmocní.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{x}_j)^2}$$

x_j jsou hodnoty z namodelovaného průběhu, který chceme uložit.
$\bar{x}_j$ jsou hodnoty z původního průběhu.

Střední kvadratická chyba je tedy odchylka tzv. reziduálních (Odchylky jednotlivých naměřených hodnot od ukládaného modelu měření). [2] Na obr. 2.3 jsou zobrazeny průběhy dvou veličin, které mají rozdílný průběh, ale jejich namodelovaný průběh určený k uložení je stejný. RMSE vyjádří odchylku namodelovaného průběhu od toho původního a tudíž slouží jako indicie, jak moc se od sebe průběhy liší.



Obrázek 2.3: Ukázka signálů s RMSE 2 a 5

MSE plní stejnou funkci jako RMSE, ale vyjadřuje průměr absolutních hodnot všech odchylek.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}|x_j - \bar{x}_j|$$

MSE a RMSE jsou nejpoužívanější metriky v oblasti měření přesnosti spojitých proměnných. Obě veličiny si jsou podobné, ale každá z nich má lepší použití v různých situacích, např. RMSE se hodí lépe u měření, kde jsou důležité velké odchylky. Na druhou stranu MSE je jednodušší k interpretaci. Podrobnějším porovnáním, z kterého jsem zde čerpal, se zabývá článek [3].

2.2 Kompresní algoritmy

O ztrátové i bezeztrátové kompresi měření elektrické energie bylo napsáno již mnoho. Hojně navrhovaným způsobem komprese je vlnková transformace [4, 5, 6, 7]. Dále se touto problematikou zabýval např. de Souza a kolektiv [8], který navrhuje jako dobrou ztrátovou metodu ke komprimaci ustálených veličin rozklad na singulární hodnoty nebo Dapper a spol. demonstrující zde [9] kompresi pomocí rychlé Fourierovy transformace, polynomiální komprese a následné komprese Deflate algoritmem.

Kromě měření, jak moc dokáží kompresní algoritmy zmenšit ukládaná data, je zvolení správného kompresního algoritmu závislé i na náročnosti algoritmů na operační paměť. V [10] jsou porovnávány bezeztrátové kompresní algoritmy s ohledem na využití v embedded systémech, které mají omezenou kapacitu paměti a menší výpočetní výkon.

Velmi relevantní publikace je [11] zabývající se, podobně jako tato práce, předběžným zakódováním a následnou kompresí algoritmy Bzip2, Lzma a Deflate, kde prvotní zakódování má zvětšit následnou efektivitu kompresních algoritmů. Předběžné bezeztrátové či ztrátové zakódování by mělo zmenšit velikost souboru jako takovou nebo transformovat data do co nejhomogennějšího tvaru (vytvoření co nejvíce za sebou zřetězených jedniček či nul), protože taková data se kompresním algoritmům lépe komprimují. Další práce [12], která se zabývá touto problematikou, testuje účinky delta kódování, PPM metody [13] a Time based bezeztrátového kódování na kompresi dat z chytrých sítí pomocí Bzip2 a Lzma. V obsáhlé zprávě [7] Tcheou a kolektiv popisují velké množství v průmyslu využívaných druhů komprese včetně Lzma, Bzip2 a různých druhů ztrátového i bezeztrátového kódování.

2.2.1 Bzip2

Bzip2 je open source kompresní program vytvořený Julianem Sewardem v roce 1996. Nejnovější verze programu vyšla 20. září 2010. Tento algoritmus dosahuje větší efektivity komprese než Deflate nebo LZW, ale zároveň je pomalejší. Bzip2 komprese trvá výrazně déle než Bzip2 dekomprese.[14]

Publikaci o bezeztrátovém kódování a kompresi dat se záznamy kvality elektrické energie publikoval v roce 2009 tým z TUL.[15] Bzip2 zde v porovnání s Huffmanovým, aritmetickým a MiniLZO kódováním dosahuje nejlepšího CR jak při ukládání nezakódovaných čísel, tak při ukládání diferenciálně kódovaných dat.

Postup Bzip2 algoritmu při kompresi:

1. Run-length kódování (RLE)[16]
2. Burrowsova–Wheelerova transformace[17]
3. Move-to-front transformace[18]
4. Znovu Run-length kódování
5. Huffmanovo kódování[19]

6. Vybrání vhodného Huffmanova stromu
7. Zakódování Huffmanova stromu do jedničkové soustavy
8. Delta zakódování bitových délek Huffmanova kódu a bitového pole obsahujícího použité symboly

Při dekompresi je aplikován stejný postup, ale v opačném směru.[14]

2.2.2 Lzma

Dalším bezeztrátovým algoritmem je Lempel-Ziv-Markov chain algoritmus. Vyvíjen mezi lety 1996 až 2001, Lzma byl v roce 2001 přidán do programu 7-zip. V roce 2004 byla publikována první verze Lzma source development kit (LZMA SDK). Nejnovější verze 18.06 vyšla 30.12. 2018. Jedná se o vylepšenou verzi LZ77 algoritmu upraveného tak, aby se maximálně zvětšil stupeň komprese, udržovala se vysoká dekompresní rychlost i nízké paměťové nároky pro dekompresi.[20] Podrobná dokumentace s popisem algoritmu není k dispozici, ale slovy autora je algoritmus realizován kombinací LZ77, Markovových řetězců a Intervalového kódování (Range Coder).[21]

Kraus a kolektiv ve své práci zabývající se využitím agregačních, polynomiálních a Spline modelů s různými kompresními algoritmy zde [22] testovali Huffmanovo, aritmetické, intervalové kódování, ZIP, GZIP, Bzip2 a Lzma. Poslední zmíněné je vyhodnoceno jako nejefektivnější.

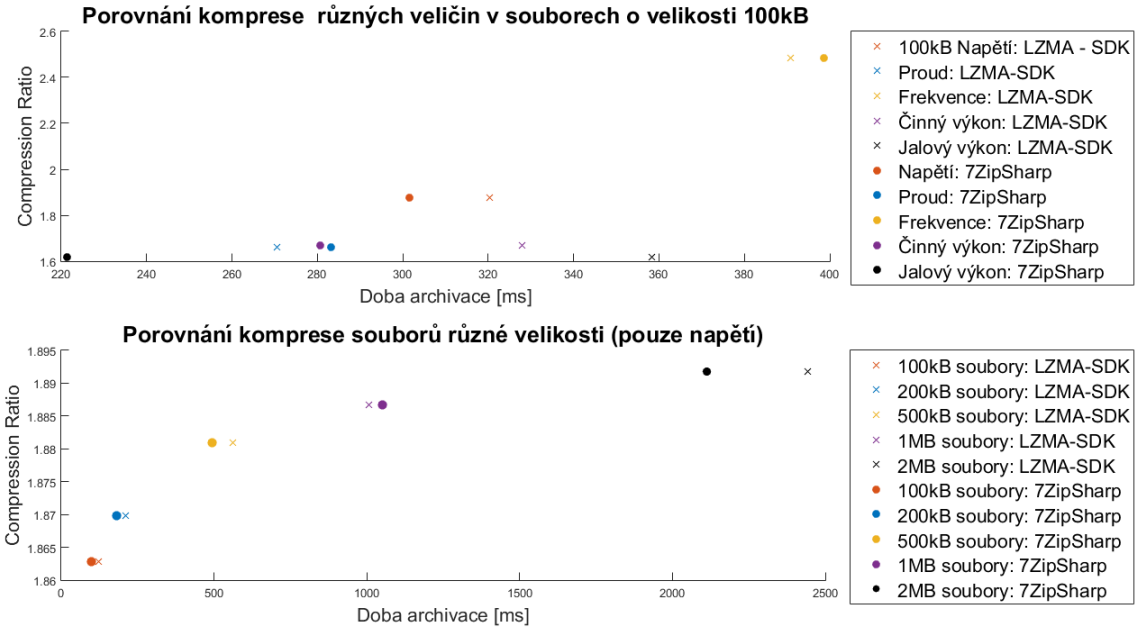
2.2.3 Deflate

Knihovna Zlib byla napsána Jean-loup Gaillyem a Markem Adlerem v roce 1995. Nejnovější verze 1.2.11 vyšla 15. ledna 2017. Je abstrakcí Deflate algoritmu, který je nejpoužívanějším algoritmem pro archivní formát ZIP. Souborový formát Gzip (s .gz příponou) také využívá Deflate kompresi a přidává ke komprimovaným souborům tzv. CRC data[23], která pomáhají odhalit poškození dat. Samotný Deflate algoritmus je kombinací LZ77[24] a Huffmanova kódování[19]. Předběžný algoritmus zlepšující následnou kompresi Deflate algoritmu je publikován v [25].

2.2.4 Implementace algoritmů a úvodní porovnání

Při rozhodování, jakou implementaci **LZMA SDK** použiji k testování, jsem se rozhodoval především mezi knihovnami SevenZipSharp a LZMA-SDK. Obě dvě jsou dostupné jako nuget package ve Visual Studiu, kde jsou nejpoužívanějšími balíčky pro práci s Lzma.

Kvůli porovnání jsem obě knihovny testoval na 113 CEA souborech a zaznamenával jejich stupeň komprese a časovou náročnost. V prvním testu jsem z každého CEA souboru exportoval měření do binárního souboru a archivoval ho pomocí obou knihoven. V druhém testu jsem exportoval jednu veličinu (průměrné napětí) do binárních souborů různé velikosti a zaznamenával jsem výše uvedené vlastnosti.



Obrázek 2.4: Porovnání LZMA-SDK a SevenZipSharp

Pro přehlednost přikládám pouze graf s průměrnými hodnotami měření. Obě knihovny měly v podstatě totožný stupeň komprese napříč všemi veličinami. Časová náročnost byla podobná u souborů s měřením napětí, frekvence a proudu, nicméně u výkonu dosáhlo LZMA-SDK výrazně horších výsledků než SevenZipSharp. V několika případech měla knihovna LZMA-SDK lepší výsledky, ale většinou bylo rychlejší SevenZipSharp. LZMA SDK je volně stažitelná i jako C# kód od autora Lzma. Tato implementace Lzma byla v mém programu výrazně méně neefektivní z hlediska časové náročnosti a komprese s ní trvala oproti SevenZipSharp v průměru více než dvakrát déle. Proto jsem se k testování rozhodl použít SevenZipSharp.

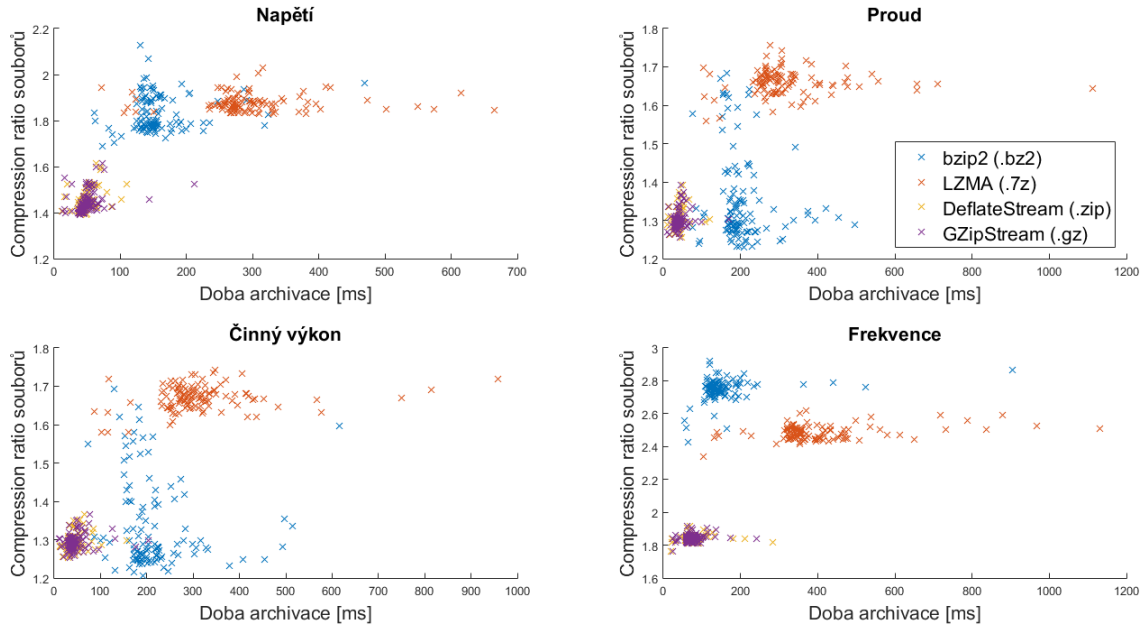
Dvě nejpoužívanější knihovny pro .NET platformu implementující **Bzip2** kompresi jsou SharpZipLib a SharpCompress. Podobně jako u Lzma, i zde měly knihovny velmi podobné výsledky, nicméně SharpCompress měl problém s kompresí malých souborů pod 1MB a výsledné bz2 soubory byly prázdné. Vzhledem k funkčnosti SharpZipLib jsem se implementací SharpCompress více nezabýval a pro testování jsem zvolil SharpZipLib.

.NET Framework obsahuje několik tříd určených ke kompresi souborů. První je DeflateStream, která realizuje **DEFLATE** algoritmus. Druhá třída, GzipStream, je téměř stejná jako DeflateStream, ale na rozdíl od ní generuje GZipStream i některá metadata. Pokud data ukládáme jako .gz soubor, některé kompresní nástroje jako např. Winrar mohou tento soubor dekomprimovat nebo s ním jinak pracovat.[26] Od verze 4.5 používá .NET k realizaci těchto tříd **zlib** knihovnu.[27] Jelikož mohu De-

flate algoritmus realizovat oběma způsoby, zkusím obě knihovny otestovat a posoudit, kterou je výhodnější použít.

Ze 113 CEA souborů exportuji napětí, proud, frekvenci a činný výkon do binárních souborů, které zkomprimuji metodami, které jsem vybral. Zaznamenám časovou náročnost s stupeň komprese. Ověřím funkčnost mého testovacího kódu, porovnám výše zmíněné metody a kompresní algoritmy při práci se soubory velkými zhruba 340kB. (Velikost je dána počtem naměřených hodnot uložených v jednom CEA souboru.)

GZipStream tvořila vždy o 18 bajtů větší velikost souboru než DeflateStream. Kromě těchto marginálních dat byly soubory totožné, což se projevilo v testech téměř stejnými výsledky GZipStreamu a DeflateStreamu jak ve stupni komprese, tak v časové náročnosti (viz 2.5). Kvůli metadatům měly .zip soubory nepatrně lepší CR i časovou náročnost (pohybující se na úrovni statistické chyby). Jelikož já tato data nepotřebuji, budu používat k testování Deflate algoritmu třídu DeflateStream.



Obrázek 2.5: Úvodní porovnání kompresních algoritmů

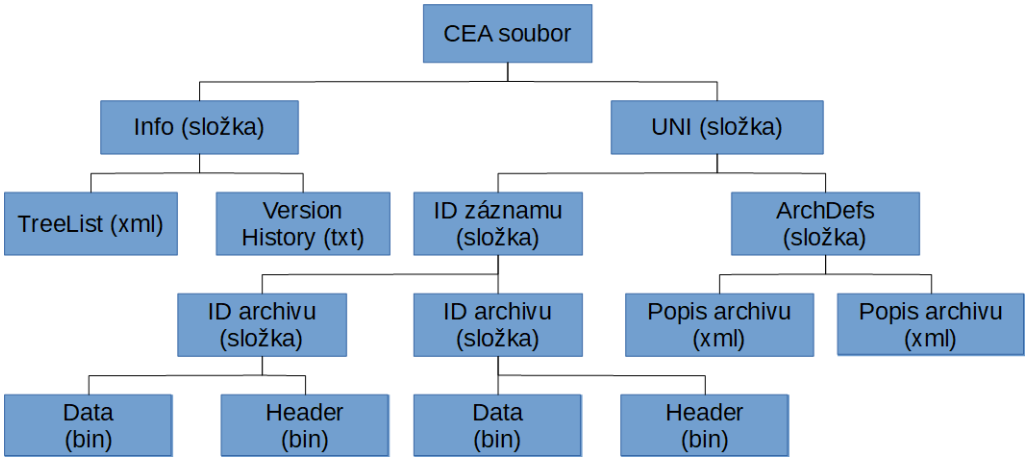
LZMA měl v průměru nejlepší CR při kompresi souborů s proudem, činným výkonem a napětím. Zároveň měl ale největší nároky na čas. Bzip2 se choval různorodě. Při kompresi frekvence dosahoval jasně největšího CR s malou časovou náročností, u napětí skončil těsně druhý, ale u činného výkonu a proudu se jeho stupeň komprese propadl na podobnou úroveň jako Deflate (v průměru byl sice o 4% až 6% nad ním, ale jeho výkon byl velmi nekonzistentní). Deflate algoritmus komprimoval nejrychleji, nicméně jeho CR byl nejmenší.

3 Archiv a archivované veličiny

3.1 CEA soubory

Posloupnosti měření elektrických veličin jsou uloženy v CEA souborech identifikovatelných .cea příponou. Podrobnému popisu struktury CEA souborů se věnuje Jan Moravec ve své bakalářské práci Zpracování a vizualizace dat analyzátorů v OS Android.[28] CEA soubory mají strukturu několika složek a souborů komprimovaných pomocí zip komprese. Struktura archivu je načrtnuta na obrázku 3.1. Měření jsou uložena binárně. Tento způsob ukládání dat je obvyklý také uvnitř mikroprocesorových měřících a monitorovacích zařízení.[29]

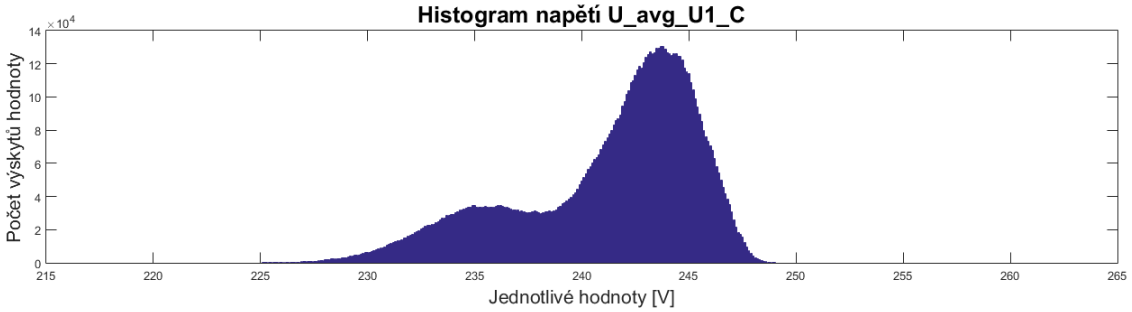
Z těchto důvodů budu exportovat naměřená data z CEA do binárních souborů a tyto soubory různým způsobem testovat.



Obrázek 3.1: CEA soubor

3.2 Napětí

V CEA archivu se vyskytují napětí dosahující několika stovek Voltů, vetšinou kolísající kolem 230-245V nebo 400V. Průběhy jsou ustálené na konstantní hodnotě, kolem které kolísají a jejich velikost se příliš nemění. Jak můžeme vidět v obrázku 3.2, hodnota jednoho napětí (U_avg_U1) se ve 113 CEA souborech pohybovala v naprosté většině případů mezi 230 a 248 Volty.

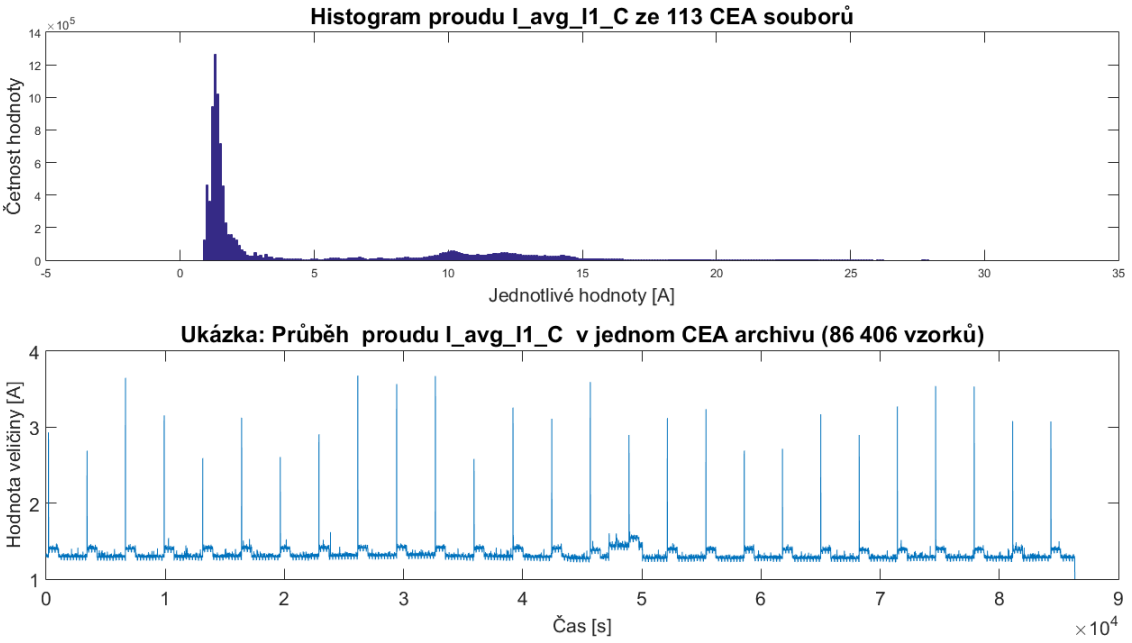


Obrázek 3.2: Histogram U_avg_U1_C ze 113 CEA souborů

3.3 Proud

Co se velikosti týče, proudy jsou velmi rozmanitá skupina. Průběh vypadá tak, že proud kolísá kolem nějaké konstanty (a tato konstanta se může měnit např. podle toho, jaká je část dne nebo jestli je víkend) a výjimečně se objeví extrém dosahující několikanásobné velikosti předešlých hodnot. Při ukládání hodnoty proudu je potom důležitý hlavně výskyt extrémů přesahujících nominální hodnotu proudu. U proudů je tedy důležitá informace, zda došlo k překonání určité hodnoty nebo zda se změnila konstanta, kolem které proud kolísá.

Na obrázku 3.3 je vidět kumulace hodnot kolem 1 Ampéru a dále menší množství extrémů, které se v tomto měření pravidelně vyskytovaly (viz ukázka průběhu proudu v 3.3). Podobný průběh se vyskytoval i u dalších měření proudu.

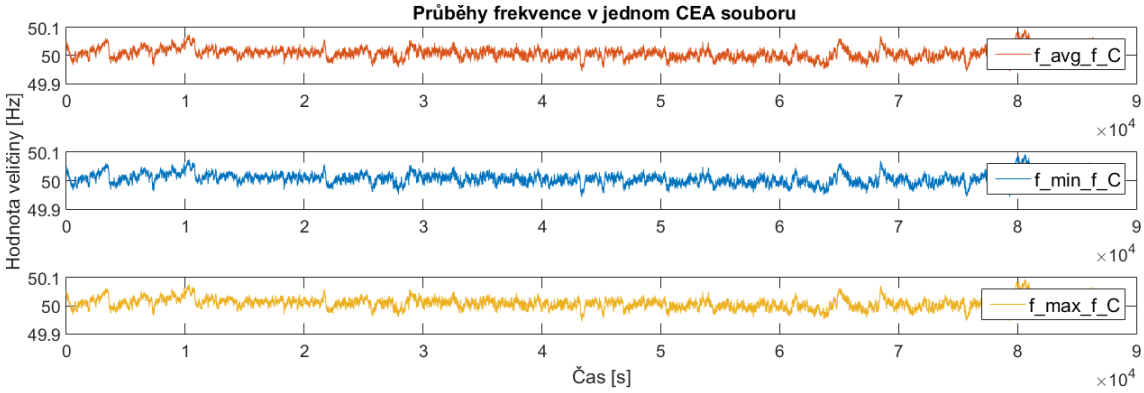


Obrázek 3.3: Histogram a ukázka průběhu I_avg_I1_C

3.4 Frekvence

CEA soubor obsahuje 3 frekvence: f avg, f max a f min. Všechny tři veličiny mají extrémně podobné průběhy a to natolik, že se v grafu překrývají (viz 3.4). Frekvence se udržují na hladině 50Hz a mají velice malé odchylky nepřesahující desetinu Hertze. (Předpokládám, že rozdíl byl způsoben chybou měření, nikoliv rozdílnými hodnotami).

Pro tuto veličinu je opět důležité zaznamenávat větší odchylky od konstanty (v tomto př. 50Hz), pokud vezmu výše zmíněné vlastnosti v potaz, přijde mi dostatečné tuto veličinu archivovat pouze jednou.

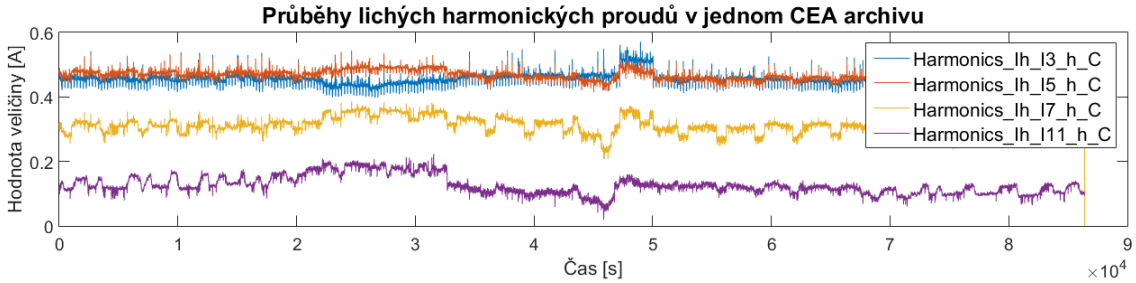


Obrázek 3.4: Průběhy f_avg_f_C, f_min_f_C a f_max_f_C

3.5 Harmonický proud

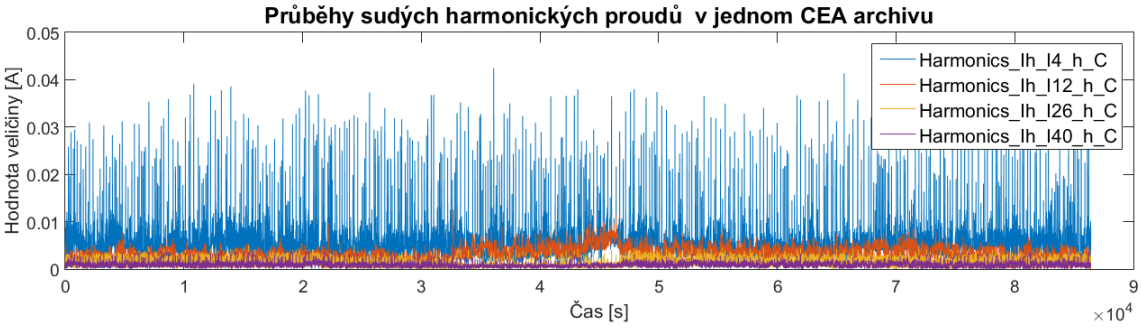
Vlastnosti harmonických proudů lichého pořadí se lišily od vlastností proudů sudých.

První lichý harmonický proud je největší (přes 1A) a má podobný tvar jako normální proud, další liché harmonické proudy se zmenšují a postupně se blíží k nule (5. harmonický: cca 0,5A, 7. harmonický: cca 0,3A, 11. harmonický: cca 0,15A, ...).



Obrázek 3.5: Ukázka průběhů lichých harmonických složek I_1

Sudé harmonické proudy sice s lichými sdílejí zmenšování velikosti s pořadovým číslem, tato vlastnost je zde ale nepatrná, jelikož hned od prvních složek jsou jejich velikosti téměř nulové. Druhý harmonický se pohybuje okolo 0,02A a další jsou ještě menší.

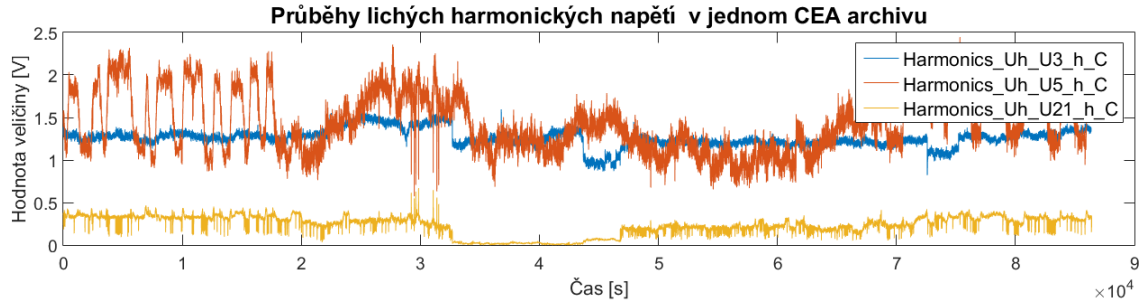


Obrázek 3.6: Ukázka průběhů sudých harmonických složek I_1

3.6 Harmonické napětí

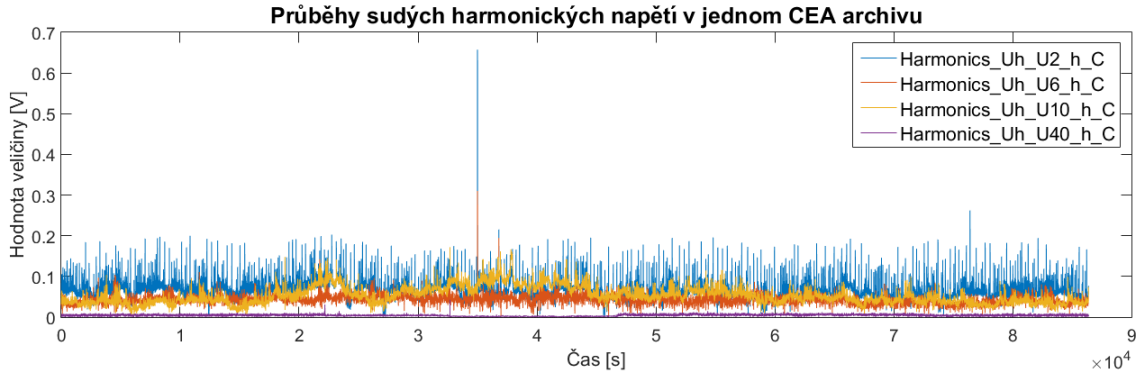
Vlastnosti harmonických napětí lichého pořadí se opět lišily od vlastností napětí sudých.

První liché harmonické napětí je největší (přes 240V) a má téměř stejný průběh jako nominální napětí. Opět je dle mého názoru na místě uvažovat o uchovávání pouze jedné z těchto veličin. Další harmonická napětí se nepravidelně zmenšují a blíží nule (U3: cca 1,3V, U21: cca 0,4V; U40: cca 0,07V).



Obrázek 3.7: Ukázka průběhů lichých harmonických složek U_1

U sudého harm. napětí se opakuje efekt popsaný u sudých harmonických proudů. Hodnoty jsou sestupné a výrazně menší než liché harmonické (v řádech desetin a setin Voltu).

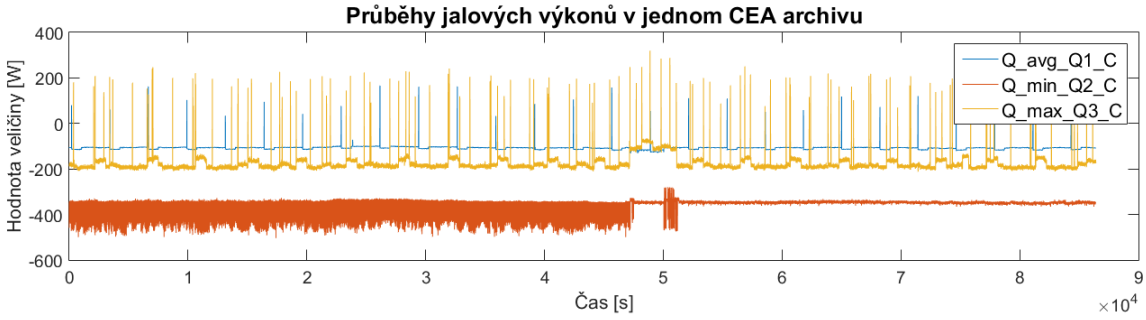


Obrázek 3.8: Ukázka průběhů sudých harmonických složek U_1

3.7 Výkon

CEA soubory obsahují měření činného i jalového výkonu.

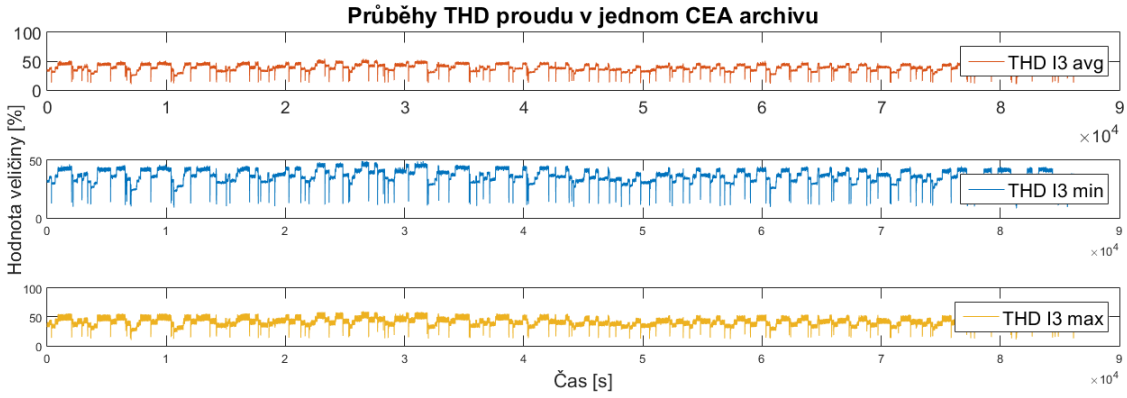
Na rozdíl od ostatních veličin obsahoval jalový výkon kromě kladných i záporné hodnoty. Vyskytovaly se velké extrémy v řádech stovek Wattů a některé jeho průběhy byly velmi proměnné. Průběhy měly často podobný tvar jako průběhy proudů. Příkladem je P1_avg_P1 a Q1_avg_Q1, které měly, stejně jako proud I1 (viz obr. 3.3), opakující se extrémy v jinak velmi ustáleném průběhu. Hodnoty se pohybovaly v řádu stovek až tisíc Wattů. (Jalový výkon byl často v jediném měření tvořen zápornými hodnotami s extrémy v kladných hodnotách.)



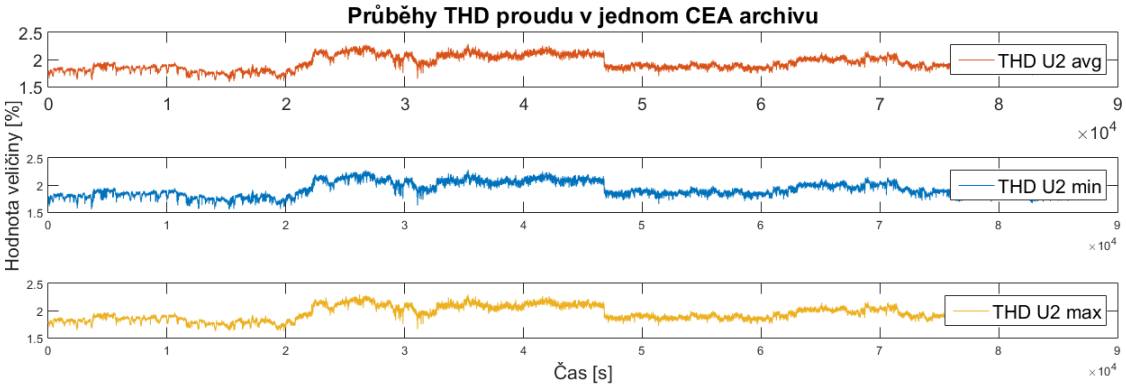
Obrázek 3.9: Ukázka průběhů jalového výkonu

3.8 THD (celkové harmonické zkreslení)

THD proudu a napětí se vyskytovaly ve trojicích: průměrná, maximální a minimální hodnota. Tyto grafy měly velmi podobné průběhy, které měly od sebe určitý offset a z velké části se překrývaly (viz 3.10). U proudu se hodnoty pohybovaly v řádu desítek procent, u napětí v jednotkách (viz 3.11).



Obrázek 3.10: Ukázka tří průběhů THD proudu



Obrázek 3.11: Ukázka tří průběhů THD napětí

3.9 Shrnutí

- V CEA souborech se nachází velké množství různých měření napětí, proudu, činného a jalového výkonu, frekvence, harmonických složek napětí a proudu a THD napětí a proudu.
- Soubory nemají konstantní množství měření, tudíž i jejich velikosti jsou různé.
- Zatímco některé veličiny, jako frekvence či napětí, mají většinou ustálené průběhy, jiné veličiny mají průběhy často velmi proměnlivé.
- Naměřené hodnoty mají proměnlivý počet cifer a desetinných míst.
- Obsah archivů je tedy značně rozmanitý a vhodné kompresní algoritmy by tedy buď měly být schopné efektivně komprimovat velmi rozdílné druhy dat, nebo je třeba data předběžně zakódovat tak, aby se co nejvíce homogenizovala.

4 Komprimace souborů různé velikosti

První série testů se zabývá kompresí binárních souborů různé velikosti obsahujících naměřené hodnoty elektrických veličin. Pro testování jsem použil vzorek 113 CEA souborů. Jedno měření v testovaných CEA souborech obsahovalo kolem 86 tisíc hodnot, tudíž při exportu do binárního souboru jako Single (4B) hodnoty měl soubor velikost kolem 340kB. Na tomto kvantitativním rozdělení ale není třeba lpět a pro efektivnější uchování měření se může osvědčit jiná velikost, která bude po kompresi některým kompresním algoritmem dosahovat většího stupně komprese. Měření jednotlivých veličin exportuji do stejně velkých binárních souborů. Testovány budou tyto veličiny: Napětí, proud, frekvence, činný a jalový výkon, harmonické napětí a proud, THD napětí a proudu. Testy by měly pokrýt ty nejdůležitější elektrické veličiny vyskytující se v archivech.

Binární soubory archivuji pomocí Lzma, Bzip2 a Deflate. Zaznamenal jsem dobu archivace a stupeň komprese [velikost před kompresí/ velikost po kompresi].

Všechny výše zmíněné testy provádím i pro několik různých způsobů zápisu měření. Nejprve otestuji kompresi nezakódovaných měření, následují testy delta kódování a pevné řádové čárky.

4.1 Ukládání hodnot jako Single

4.1.1 Popis testů

Jelikož mají hodnoty v archivech čísla za desetinnou čárkou, je pro normální zápis možné volit z datových typů Double nebo Single. Double (8B) je zbytečně velký a proto je nejlepším řešením ukládat hodnoty bez zakódování jako Single.

4.1.2 Výsledky testů

Efektivita komprese se u jednotlivých měření výrazně lišila. Nejvyššího CR algoritmy dosahovaly při kompresi frekvence. Bzip2 zde dosáhl nejvyššího stupně komprese těsně před LZMA, který byl mírně horší. Oba algoritmy si vedly podobně při kompresi napětí, nicméně při kompresi všech ostatních veličin dosahoval nejlepších výsledků LZMA. Bzip2 naopak zaostával. Bzip2 měl také trvale menší časovou náročnost než LZMA. Jako ukázku přikládám průměrné CR a čas komprese napětí.(tab. 4.1) Je zřejmé, že Bzip2 komprimoval napětí dvakrát až třikrát rychleji nez Lzma.

Tabulka 4.1: Průměrné CR a čas [CR/t] pro měření napětí

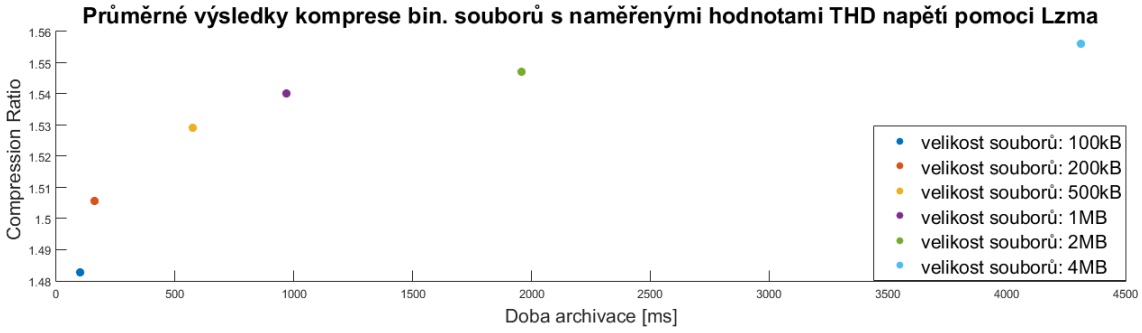
	100kB	200kB	500kB	1MB	2MB
bz2	1,78/ 87ms	1,79/ 125ms	1,87/ 255ms	1,91/ 419ms	1,92/ 863ms
7z	1.86/ 122ms	1,87/ 211ms	1,88/ 563ms	1,89/ 1s	1,89/ 2,44s
zip	1,44/ 19ms	1,44/ 37ms	1,45/ 96ms	1,45/ 162ms	1,44/ 346ms

Krom výjimek dosahoval Lzma největšího stupně komprese. U veličin pohybujících se kolem nuly či v jednotkách a u veličin obsahujících větší extrémy byl Lzma výrazně lepší než Bzip2 i Deflate a na rozdíl od Bzip2 měl dobré výsledky v podstatě u všech druhů dat. Bzip2 se při aplikaci na malá nebo proměnlivá data viditelně zhoršoval a např. u 40. harmonického napětí, které kolísalo kolem nuly, dosahoval stejného CR jako Deflate, který měl trvale nejmenší dosažené CR. Největší výhoda Deflate byla jeho velká rychlost. Jako ukázka poslouží tabulka výsledků páté harmonické (tab. 4.2), která měla velikost zhruba 0,45A.

Tabulka 4.2: Průměrné CR a čas [CR/t] pro 5. harmonický proud

	100kB	200kB	500kB	1MB	2MB
bz2	1,32/ 59ms	1,29/ 202ms	1,27/ 334ms	1,26/ 428ms	1,26/ 977ms
7z	1.54/ 68ms	1,55/ 244ms	1,56/ 504ms	1,57/ 714ms	1,57/ 1,7s
zip	1,26/ 9ms	1,26/ 32ms	1,26/ 67ms	1,26/ 97,8ms	1,26/ 231ms

Při změně velikosti souborů docházelo k malým změnám v CR. Největší odezvu měl Bzip2 na frekvenci, kdy se mezi 100kB a 4MB soubory CR zvětšilo z 2,46 na 2,95. Jinak se CR měnil minimálně a v případě Bzip2 často i záporně. Stupeň komprese Lzma při zvětšování velikosti souboru pravidelně rostl, ale opět velmi nevýrazně. Ani s využitím Deflate se neprojevovaly velké změny. Ke změně navíc docházelo u menších velikostí, tedy rozdíly mezi kompresí souborů o velikosti 100 a 200kB byly větší, než rozdíly mezi soubory s velikostí 1 a 4MB (viz 4.1.2).



Obrázek 4.1: Průměrné výsledky komprese bin. souborů s naměřenými hodnotami THD napětí pomocí Lzma

4.2 Ukládání hodnot jako Integer s pevnou řádovou čárkou

4.2.1 Popis kódování

Jelikož měření elektrických veličin tvoří desetinná čísla, nejjednodušší způsob jejich reprezentace v souborech je datový typ Single (4B). K ukládání hodnot ale lze použít i některé z celočíselných datových typů. Při ukládání menších čísel blížících se nule nebo při archivaci hodnot vyžadujících přesnost na desetinná místa je ale nevhodné ztrácet velkou část informace zaokrouhlováním hodnoty veličiny na celé číslo. V takovém případě můžeme několik cifer čísla přiřadit číslům za desetinnou čárkou a ukládat je jako fixed point. Podrobnějším popisem se zabývá sekce 2.1.4 v rešerši.

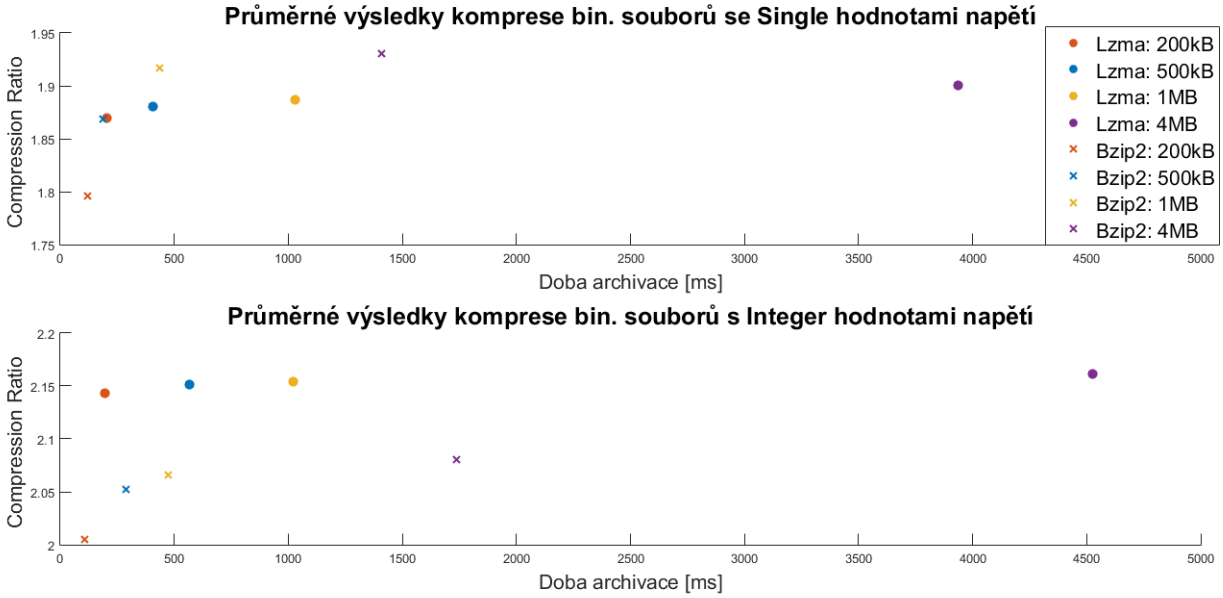
4.2.2 Popis testů

Cílem těchto testů bude porovnání komprese měření uložených do bin. souborů jako Single a jako Int32. V obou případech budou čísla zaokrouhlena na 4 desetinná místa (Hodnota 1,23456 bude exportována jako 1,2346 Single a 12346 jako Integer), dojde tedy k uložení stejného množství informace. Cílem testu je porovnat tyto způsoby kódování. Opět budu testy provádět na vzorku 113 CEA souborů. U obou kódování porovnám stupeň komprese a časovou náročnost jednotlivých komprimačních algoritmů při jejich aplikaci na binární soubory různé velikosti.

K ověření testů otestuji i ukládání hodnot jako Single a Integer s jednou desetinnou čárkou, abych mohl lépe určit chování kompresních algoritmů při různých stupních komprese a pokusím se o analýzu, jaký má tato změna efekt na CR a časovou náročnost.

4.2.3 Výsledky testů

Obecně se dá říci, že CR se při fixed point kódování zvětšilo. Největšího zvětšení dosahoval Lzma, který předčil Bzip2 i Deflate u všech veličin kromě frekvence. Jako ilustrační graf poslouží graf se soubory s napětím, jelikož v minulých testech byly u této veličiny algoritmy Lzma a Bzip2 poměrně vyrovnané. Na obrázku 4.2 je vidět kromě zvětšení CR u obou algoritmů i změna mezi Bzip2 a Lzma, kterému se stupeň komprese zvýšil o něco víc.



Obrázek 4.2: Single vs fixed point Integer: Napětí

Podrobné změny v CR a čas. náročnosti komprese jsou vyčísleny v tabulce 4.3.

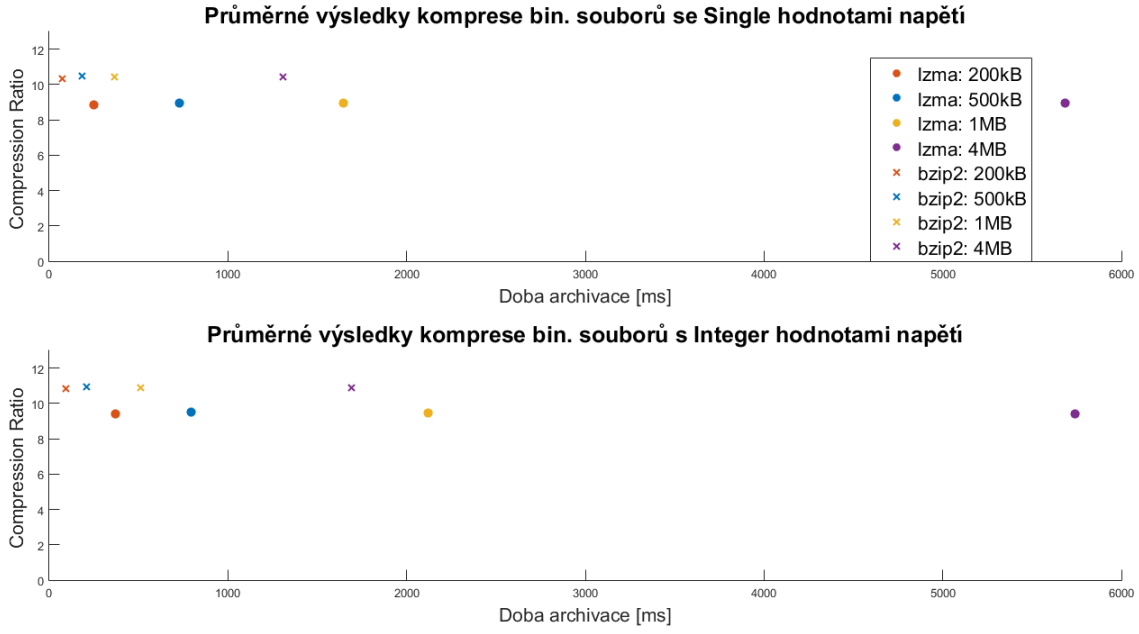
Tabulka 4.3: Poměry průměrů CR a časové náročnosti: soubory s INT/soubory se Single. Číslo před lomítkem značí CR souborů s Integery a souborů se Singly. (Pokud je číslo >1, stupeň komprese se ukládáním Integerů místo Singlů zvětšil.) Číslo za lomítkem značí poměr časové náročnosti komprese souborů s Integery a souborů se Singly (Tedy pokud je číslo >1, soubory s Integery měly větší časovou náročnost než soubory se Singly).

	200kB	500kB	1MB	4MB
Napětí: .bz2	1.12/ 0.9	1.0984/1.54	1.078/ 1.09	1.078/ 1.234
Napětí: .7z	1.14/ 0,95	1,1439/1.386	1.1417/0.991	1.1373/ 1.15
Napětí: .zip	1.13/ 1.64	1.13/ 2.23	1.1329/ 1.71	1.1325/ 1.925
Proud: .bz2	1.1827/ 1	1.1475/ 1.42	1.1351/ 1.19	1.128/ 2.189
Proud: .7z	1.2801/ 1.47	1.26/ 1.993	1.2546/ 1.662	1.2381/ 2.962
Proud: .zip	1.2012/ 2.4	1.1998/ 2.69	1.2019/ 2.56	1.2015/ 4.769
Frekv.: .bz2	1.1545/ 1.02	1.0996/ 0.97	1.0914/ 1.03	1.0799/ 2.265
Frekv.: .7z	1.2409/ 1.46	1.1865/ 1.36	1.1636/ 1.467	1.1342/ 3.77
Frekv.: .zip	1.0473/ 1.72	1.0418/ 1.59	1.0398/ 1.67	1.0384/ 4.279
Činný v.: .bz2	1.02/ 0.83	0.988/ 1.07	0.968/0.81	0.963/ 1.808
Činný v.: .7z	1.05/ 0.83	1.045/ 0.96	1.0435/ 0.8	1.0435/ 1.927
Činný v.: .zip	1.0459/ 1.09	1.0444/ 1.26	1.0433/ 1.18	1.0431/ 1.835
Jalový v.: .bz2	1.0518/ 0.83	1.012/ 1.08	0.995/ 0.86	0.991/ 1.53
Jalový v.: .7z	1.1309/ 0.82	1.1164/ 0.96	1.1102/ 0.805	1.1038/ 1.254
Jalový v.: .zip	1.1162/ 1.4	1.1132/ 1.53	1.1112/ 1.43	1.1105/ 2.605

Soubory s hodnotami proudu jako Integery dosahovaly oproti souborům se Singly o 10-20% většího CR při kompresi pomocí Bzip2, o 20% lepší u Deflate a mezi 20 a 30% při kompresi pomocí Lzma. Činný a jalový výkon P1 a Q1 mají podobný tvar průběhu jako proud I1, ale pohybují se ve stovkách Wattů (o 2 cifry větší čísla). To se projevilo u Bzip2 zhoršením CR u souborů s Integery na zhruba stejnou úroveň jako u souborů se Singly. Lzma měl zdaleka největší zlepšení CR (u proudu 28-23%, u frekvence 24-13%, činný výkon 5%, jalový výkon 10-13%) a oba další algoritmy předčil. Stupeň komprese Deflate algoritmu se s použitím fixed point zlepšil také, jeho CR ale zůstal nejhorší.

Používání Integerů ovlivnilo časovou náročnost spíše negativně. Deflate algoritmus kromě pravidelného zvětšení CR zaznamenal i zvětšení času komprese. Toto zhoršení bylo největší u souborů o velikosti 4MB. Změna časové náročnosti u Bzip2 a Lzma byla značně nelineární, v některých případech se zvětšila, někdy zmenšila. Hlavním znakem časové délky, který lze dobře pozorovat z tabulky 4.3 i z grafů 4.2 a 4.3 je velké zhoršování časové náročnosti u souborů větší velikosti (4MB).

Při ukládání pouze jednoho desetinného místa namísto čtyř se rozdíly v CR mezi soubory s Integery a soubory se Singly nezměnily. Opět docházelo k zvětšení CR. U napětí 4-5% při kompresi Bzip2 a 6% při použití Lzma. To je oproti předešlým testům ještě menší rozdíl viz 4.3. Naopak u činného výkonu došlo k zvětšení rozdílů, kdy Lzma byl o 10% a Bzip2 o 5% lepší při ukládání Integerů. Deflate dosahoval podobných výsledků jako v předchozím testu. Dalším důležitým poznatkem je, že stejně jako při ukládání hodnot jako Single, i zde změna velikosti souborů neměla velký vliv na stupeň komprese.



Obrázek 4.3: Komprese napětí s jednou desetinnou čárkou

4.2.4 Závěr

Uchovávání měření jako Integerů mělo ve většině případů za následek zvětšení CR, v nejhorších případech podobné výsledky jako ukládání Single hodnot. Lzma dosahoval největšího zvětšení pohybujícího se od 4 do téměř 30%. CR zhoršoval nestálý průběh veličin, který nejhůře snášel Bzip2 (u činného výkonu došlo i k velmi malému zhoršení v CR oproti souborům se Single hodnotami). Deflate se v CR zlepšovalo mezi 3-20%. S lepším CR se časová náročnost zhoršovala. U veličin s malým zvětšením CR (činný proud atd.) lze pozorovat zlepšení časové náročnosti, u veličin jako frekvence, napětí a proud se čas. náročnost zhoršovala. Největší rozdíl v CR byl mezi oběma způsoby kódování u malých souborů. Se zvětšující se velikostí souboru se rozdíly v CR zmenšovaly.

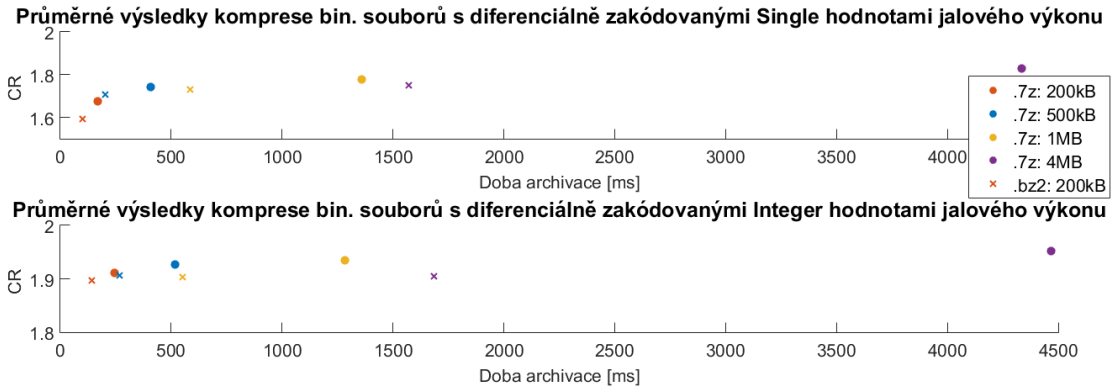
4.3 Diferenciální kódování

4.3.1 Popis

Diferenciální kódování (popsané v sekci 2.1.3 v teoretické části této práce) budu ukládat opět nejdříve jako Single se 4 desetinnými místy a poté jako Integer s prvními 4 ciframi pro desetinná místa. Takto bude možné efektivitu tohoto způsobu archivace dat přímo porovnat s předešlými testy.

4.3.2 Výsledky

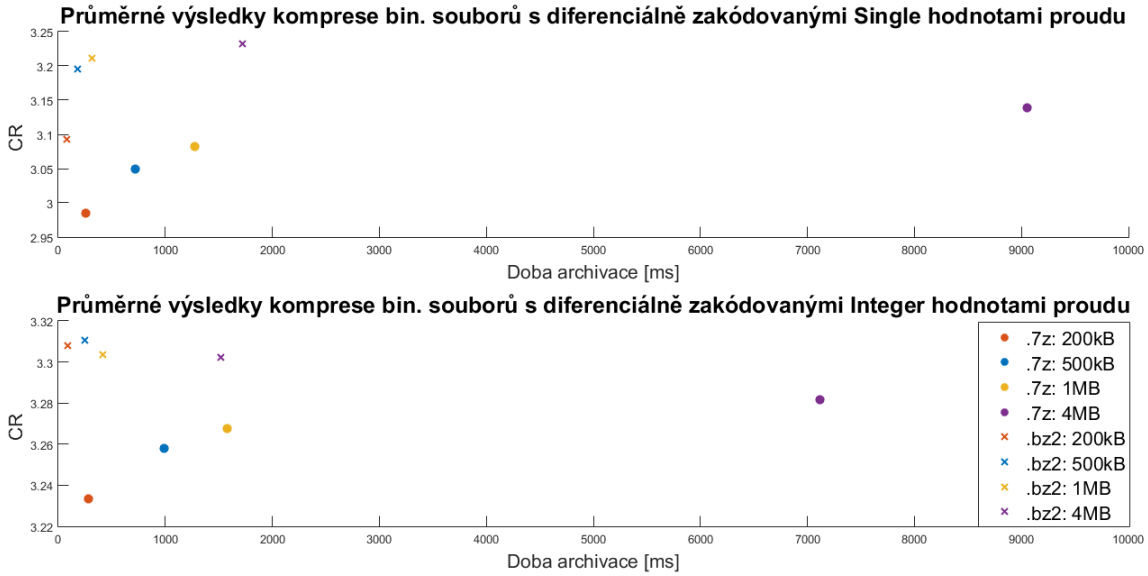
U jalového i činného výkonu a harmonických složek napětí i proudu nejlépe komprimoval Lzma (I když těsně, s max. rozdíl v CR oproti Bzip2 byl 0.1). Zároveň si ale stále udržoval největší nárok na čas. Deflate algoritmus se oproti dřívějším testům nezměnil, stále jednoznačně nejrychlejší, ale s nejmenším dosaženým CR.



Obrázek 4.4: Dif. kódovaný jalový výkon

Největšího CR dosáhly všechny algoritmy při kompresi frekvence. Stupeň komprese pro Lzma se pohyboval mezi 4.5 až 4.6 a pro Bzip2 zase mezi 4.7 až 4.85. Další

veličiny, kde měl Bzip2 lepší výsledky, byly napětí i proud. Rozdíl v CR byl zhruba 0.1 ve prospěch .bz souborů. Bzip2 také soubory zkomprimovalo v mnohem kratším čase(viz 4.5).

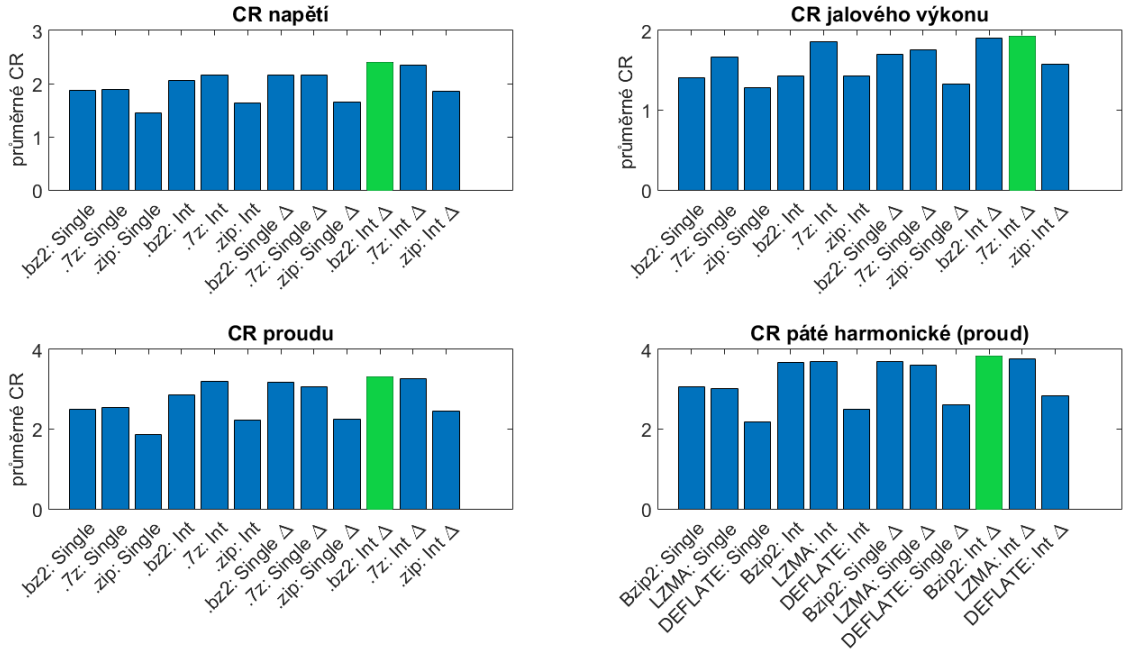


Obrázek 4.5: Dif. kódovaný proud

Ukládání hodnot jako Single i jako fixed point Integer se ukázalo jako výhodné, fixed point Integer dokonce jako nejvýhodnější způsob komprese ze všech testovaných způsobů. U všech testovaných veličin dosahovalo delta zakódování fixed point Integerů většího CR, než delta kódování Single hodnot. Bzip2 a Lzma měly vyrovnané výsledky ve stupni komprese. Frekvence, napětí, proud, harmonické složky napětí a proudu nejlépe archivoval Bzip2, činný a jalový výkon ovládl Lzma. Bzip2 byl ale trvale rychlejší než Lzma a vzhledem k tomu, že při delta kódování dokázal s Lzma držet krok v kompresi všech veličin, označil bych ho v těchto testech za nejvhodnější algoritmus.

4.4 Shrnutí

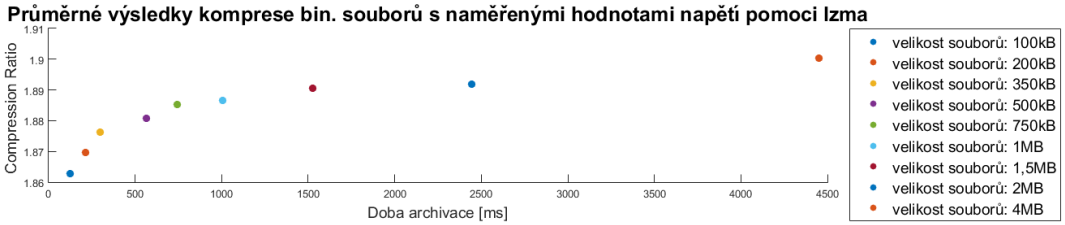
Testy provedené v této kapitole měly najít rozdíly mezi ukládáním hodnoty jako Integery a jako Singly, mezi diferenciálním kódováním a normálním zápisem měření, mezi jednotlivými algoritmy a také mezi různými velikostmi komprimovaných souborů. V ilustraci 4.6 jsem na čtyřech různých veličinách vykreslil průměrné výsledky jednotlivých algoritmů a způsobu zápisu dat.



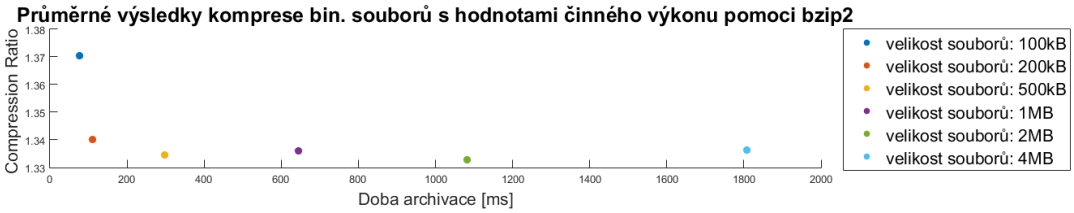
Obrázek 4.6: Průměrné výsledky testů.

- Použití diferenciálního kódování (v grafu 4.6 značeno jako Δ) se na stupni komprese podepsalo kladně, stejně tak jako ukládání hodnot jako Integer s pevnou řádovou čárkou (v grafu ozn. jako Int). Kombinace těchto dvou technik proto dosahovala v testech nejlepších výsledků. Všechny kompresní algoritmy tedy dosahovaly nejvyššího CR při použití kombinace delta kódování a pevné řádové čárky. CR kompresních algoritmů Bzip2 a Deflate se při použití této techniky zlepšil u všech veličin zhruba o 30% oproti normálnímu ukládání hodnot jako Single. CR u Lzma se také zvětšil o 15 - 30%.
- Bzip2 komprimoval s nejvyšším stupněm komprese homogenní data (např. frekvence, která vždy kolísala těsně kolem 50Hz). Zároveň komprimoval výrazně rychleji než Lzma. Dosažený CR byl ale velmi situační a při kompresi souborů s proměnlivějšími hodnotami jeho velikost výrazně klesla (někdy až pod úroveň Deflate algoritmu, viz CR jalového výkonu v 4.6).
- Lzma byl daleko všestrannější a dosahoval dobrých výsledků na všech testovaných datech za cenu vyšší časové náročnosti.

- Deflate měl opačné výsledky. Komprese souborů probíhala s přehledem nejrychleji ale s nejmenším CR.
- Stupeň komprese jako takový se při změně velikosti souboru zásadně neměnil. Největší rozdíl se vyskytoval vždy u souborů menších velikostí (100 až 500kB) a s přibývající velikostí se zmenšoval (rozdíl mezi 1MB a 4MB byl zpravidla menší než rozdíl mezi 100kB a 500kB).
- Změna ve stupni komprese u Lzma byla vždy kladná, tedy čím větší velikost souboru, tím lepší stupeň komprese (viz 4.7). Naproti tomu změna v CR u Bzip2 byla někdy kladná, jindy se CR při zvýšení velikosti souborů zmenšilo (viz 4.8).
- Vzhledem k předchozím dvěma bodům je možné určit konstantní optimální velikost souboru pouze pro Lzma. Pokud vezmu v potaz i fakt, že CEA archivy obsahují pouze malé množství měření frekvencí (většinou 3) a mnohonásobně větší množství měření napětí, proudu, výkonu a dalších veličin, Bzip2 bude mít pravděpodobně slabší výsledky, než vyplývá z obr. 4.6.
- **Za nejvhodnější kompresní algoritmus, který má nejen vysoký stupeň kom-prese, ale i stabilní chování u všech druhů dat, bych označil Lzma. Jelikož se pro Lzma stupeň komprese zvyšoval hlavně mezi 100kB a 1MB a dále už stagnoval (viz 4.7), jako nejvhodnější velikost souboru bych označil 1MB. Nakonec, jak již bylo řečeno, za nejefektivnější způsob předběžného zpracování bych zvolil kombinaci delta kódování a pevné řádové čárky při ukládání měření jako datový typ Int32.**



Obrázek 4.7: Průměrné výsledky algoritmu Lzma při kompresi napětí



Obrázek 4.8: Průměrné výsledky algoritmu Bzip2 při kompresi činného výkonu

5 Využití třídy přesnosti

Nepřesnost měření, která je vyjádřena třídou přesnosti měřícího přístroje, lze využít k transformaci ukládaných hodnot, pakliže se data budou pohybovat v povoleném pásmu stanoveném třídou přesnosti. To de facto znamená, že v určitém pásmu hodnot můžeme naměřené hodnoty zaokrouhlovat, zvětšovat i zmenšovat a ukládat je takovým způsobem, abychom zlepšili následnou kompresi některým kompresním algoritmem. V praxi není vhodné používat celé pásmo TP, abychom se vyhnuli problémům s hraničními hodnotami. Pokud se vedle sebe vyskytuje hodnota u dolní hranice pásma TP a hodnota u horní hranice pásma TP, může se stát, že rozdíl mezi těmito hodnotami bude větší, než dovolená odchylka. Tento nešvar odstraníme zmenšením povoleného pásma, ve kterém se lze pohybovat (např. na desetinu pásma třídy přesnosti). Pojem pásmo tolerance (zkr. **PT**) budu v této kapitole používat k označení části třídy přesnosti (zpravidla desetiny TP), ve které se hodnoty budou moci pohybovat.

5.1 Komprese zaokrouhlováním

Jedná se o jednoduchý druh ztrátové komprese, kterým jsem se zabýval v mém semestrálním projektu ve 2. ročníku. Snažíme se zmenšit počet míst za desetinnou čárkou na co nejmenší počet, kdy se zaokrouhlené číslo stále nachází v pásmu tolerance. Pokud se v souboru nacházejí podobně velká čísla, zaokrouhlí se v některých případech stejně. Hojnější výskyt stejných čísel, stejně tak, jako zmenšení celkového počtu cifer ukládaných čísel, povede ke zlepšení stupně komprese bezeztrátových kompresních algoritmů.

Tabulka 5.1: Porovnání zaokrouhlování dle různých PT

PT 0	PT 0,05	PT 0,1	PT 0,2	PT 1
0,9185022	0,9185	0,919	0,92	0,92
16,8826	16,88	16,88	16,9	17
1244,416	1244	1244	1244	1244

5.2 Buffer předešlých hodnot

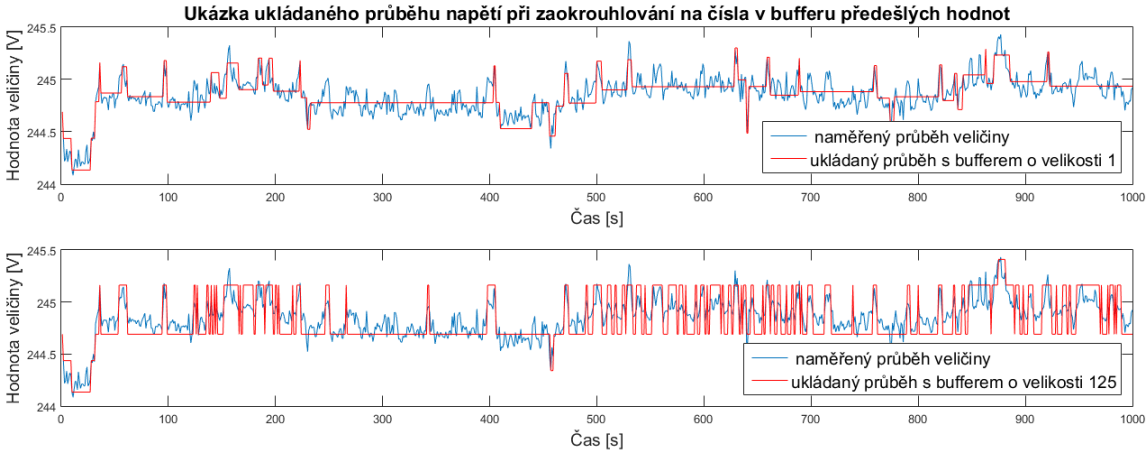
Jeden ze způsobů předběžného zpracování spočívá ve sledování, zda se velikost ukládaného čísla nedá v rámci povoleného pásma pozměnit na velikost čísla, které se v měření vyskytovalo již dříve. Tímto způsobem se zvětší počet stejných čísel nacházejících se v souboru, což zefektivní následnou kompresi bezeztrátovými algoritmy Bzip2, Lzma a Deflate. Buffer předešlých hodnot, na které se budeme snažit ukládané číslo změnit, bude realizován jako datový typ fronta. Frontu budeme procházet od konce, aby se maximalizoval počet zřetězených stejných hodnot nacházejících se vedle sebe. Při nálezu prvního čísla v bufferu, které se nachází uvnitř povoleného pásma ukládaného čísla, se velikost ukládaného čísla změní na velikost čísla nalezeného v bufferu. Následně číslo se uloží do souboru a také se přidá na konec bufferu.

5.3 Namodelované průběhy při různých parametrech

Stupeň komprese a časová náročnost se budou měnit podle změny v namodelovaném průběhu veličiny. Změna v průběhu závisí na velikosti bufferu a velikosti povoleného pásma. Jelikož se jedná o ztrátovou kompresi, je vhodné analyzovat i změnu ukládané informace od té původní v závislosti na velikosti bufferu a pásmu tolerance.

5.3.1 Pásmo tolerance 0,1

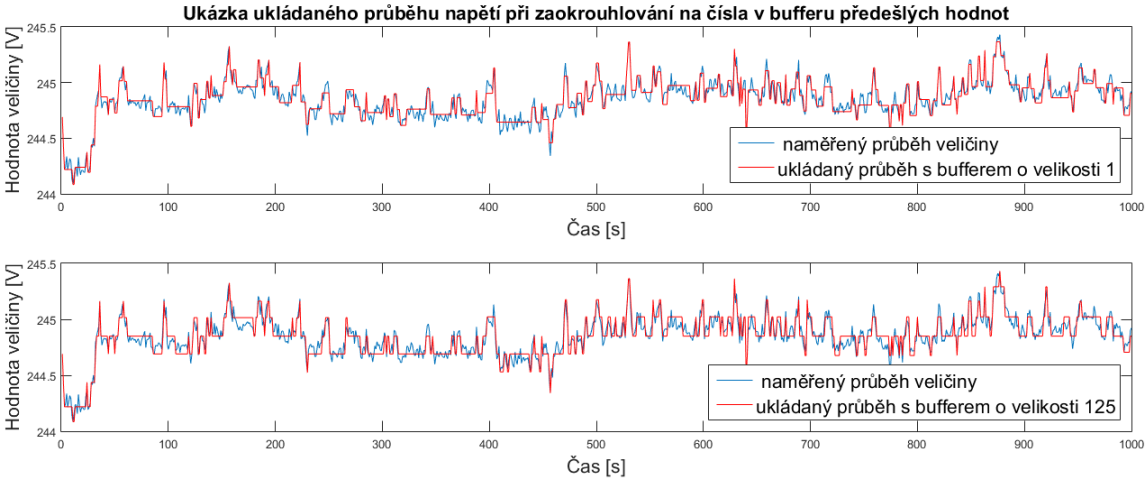
Na obr. 5.1 je vidět rozdíl v ukládaném měření při různě zvolené délce bufferu (zvolená třída přesnosti je 1 -> povolené pásmo je 0,1). Při využití bufferu o délce 125 se díky delší paměti vyskytují častěji stejné hodnoty a zároveň se zmenšuje počet unikátních velikostí veličiny, nicméně časová náročnost se díky procházení delšího bufferu zhoršuje.



Obrázek 5.1: Průběhy napětí při využití bufferů různé velikosti a PT 0,1

5.3.2 Pásmo tolerance 0,05

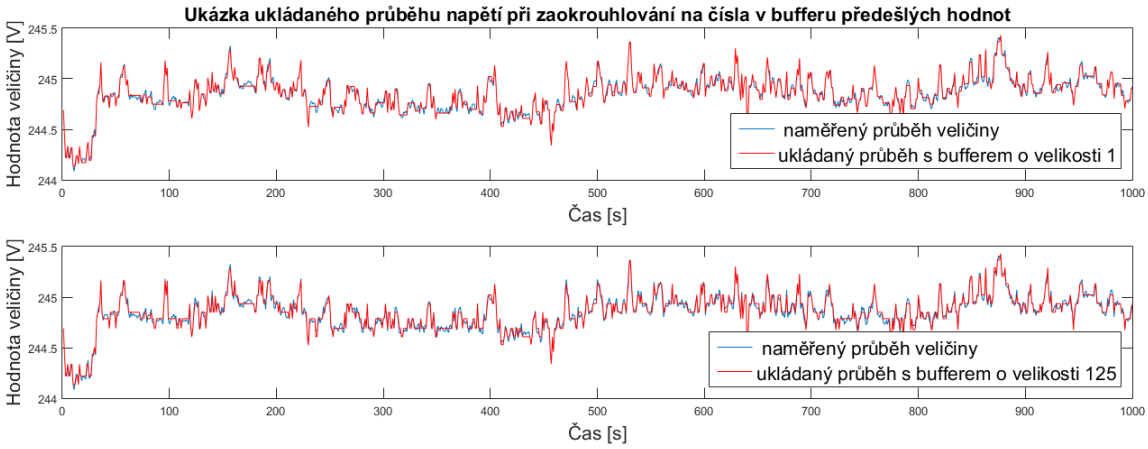
Při použití dvakrát menšího pásma tolerance dochází k nalezení vhodné hodnoty v bufferu méně často a tudíž se hodnoty v bufferu mění častěji. Průběh ve výsledku neobsahuje tolik stejných hodnot. Nejvhodnější velikost bufferu se tedy mění dle povoleného pásma. Při použití malého PT na proměnlivou veličinu se bude často bezvýsledně prohledávat celý buffer a to povede k velké časové náročnosti s pouze malým zlepšením ve stupni komprese.



Obrázek 5.2: Průběhy napětí při využití bufferů různé velikosti a PT 0,05

5.3.3 Pásmo tolerance 0,025

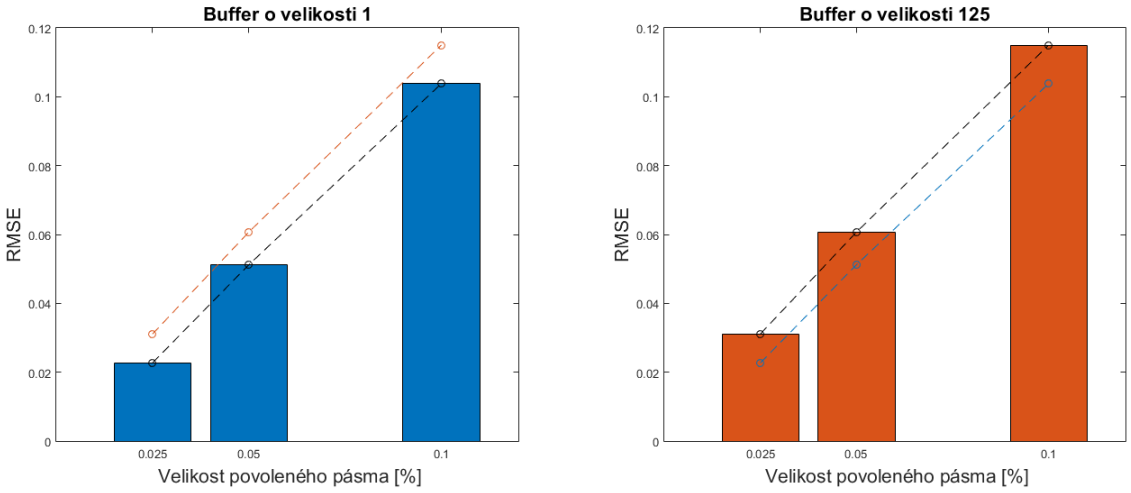
Při dalším zmenšení pásma se efekt opakuje. Řetězce za sebou jdoucích stejných hodnot jsou kratší a vzácnější, proto se při zachování stejně velké osy y a délky vzorku stává těžší a těžší vidět rozdíly mezi bufferem o vel. 1 a 125.



Obrázek 5.3: Průběhy napětí při využití bufferů různé velikosti a PT 0,025

5.3.4 Rozdíly mezi průběhy

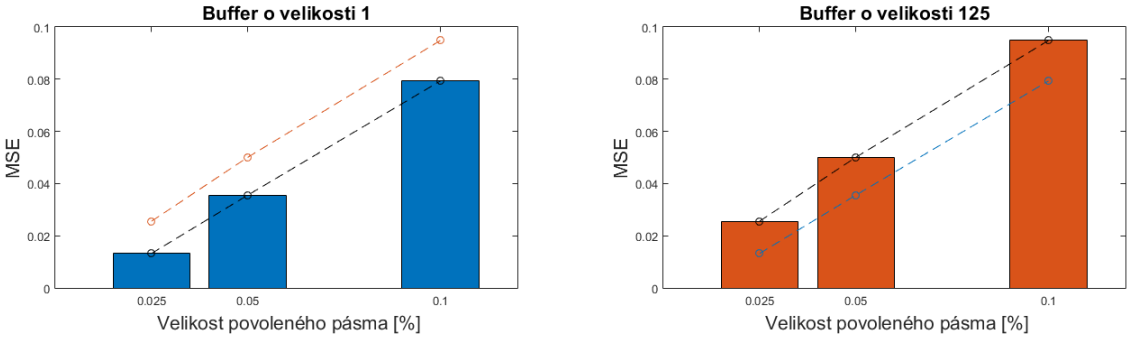
Odchylka od původního průběhu se dá popsat pomocí střední kvadratické chyby (RMSE; viz sekce 2.1.5). Na obrázku 5.4 jsou vykresleny změny této veličiny při různých povolených pásmech.



Obrázek 5.4: RMSE pro měření napětí o velikosti 86000 vzorků při využití bufferu předešlých hodnot

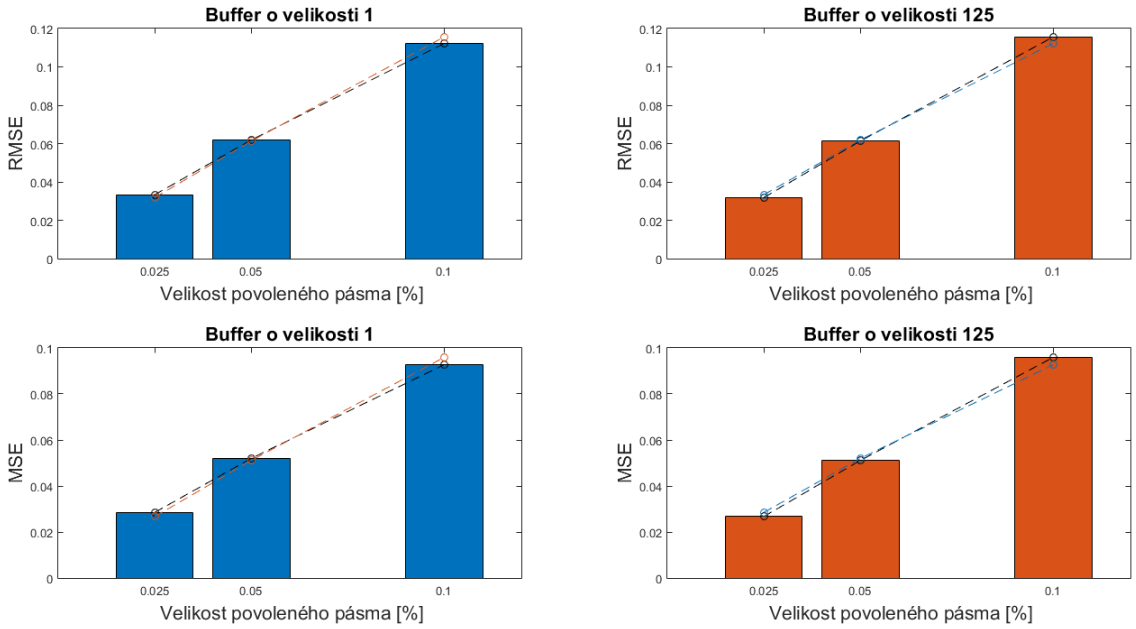
Rozdíl mezi velikostmi bufferu je znázorněn přerušovanou čárou. S pásmem přesnosti 0,1 dosahoval buffer o velikosti 125 o 0,0084 většího RMSE než při velikosti 1. Tento rozdíl se s rostoucím pásmem tolerance lineárně zvyšoval a při PT 0,025 dosahoval rozdíl v RMSE již 0,0112.

Další ukazatel použitelný k analýze odchylky od původního průběhu je střední absolutní odchylka (MSE; viz sekce 2.1.5). Pro stejný průběh je MSE vynesena v grafu 5.5. Výsledek je v tomto případě podobný RMSE: Při zvětšení pásma tolerance dochází k lineárnímu zvětšení střední absolutní odchylky. Stoupání MSE je u bufferu s velikostí 125 mírně strmější než u bufferu o velikosti 1.



Obrázek 5.5: MSE pro měření napětí o velikosti 86000 vzorků při využití bufferu předešlých hodnot

Pokud spočítáme RMSE a MSE pouze pro ty hodnoty, které byly upravené (tedy ty, které se oproti originálnímu měření změní), rozdíly mezi buffery se zminimalizují, viz obr. 5.6.



Obrázek 5.6: MSE a RMSE pouze pro upravované hodnoty při využití bufferu předešlých hodnot

Důvodem je princip, na kterém buffer funguje: v bufferu s větší kapacitou se podaří častěji najít vhodnou hodnotu a tudíž se oproti originálu změní větší množství hodnot. Takto vzniká odchylka od originálu a buffery s větší velikostí tedy budou mít i větší odchylku (jak RMSE, tak MSE). Pokud ale měříme odchylky pouze na změněných datech, poměr upravených a neupravených hodnot je irelevantní. Proto lze v grafu 5.6 pozorovat mezi buffery pouze minimální rozdíly. Velikosti RMSE a MSE se podobají velikostem stejných veličin u bufferu s velikostí 125, protože tam se častěji upravovaly hodnoty.

5.4 Popis testování CR a časové náročnosti

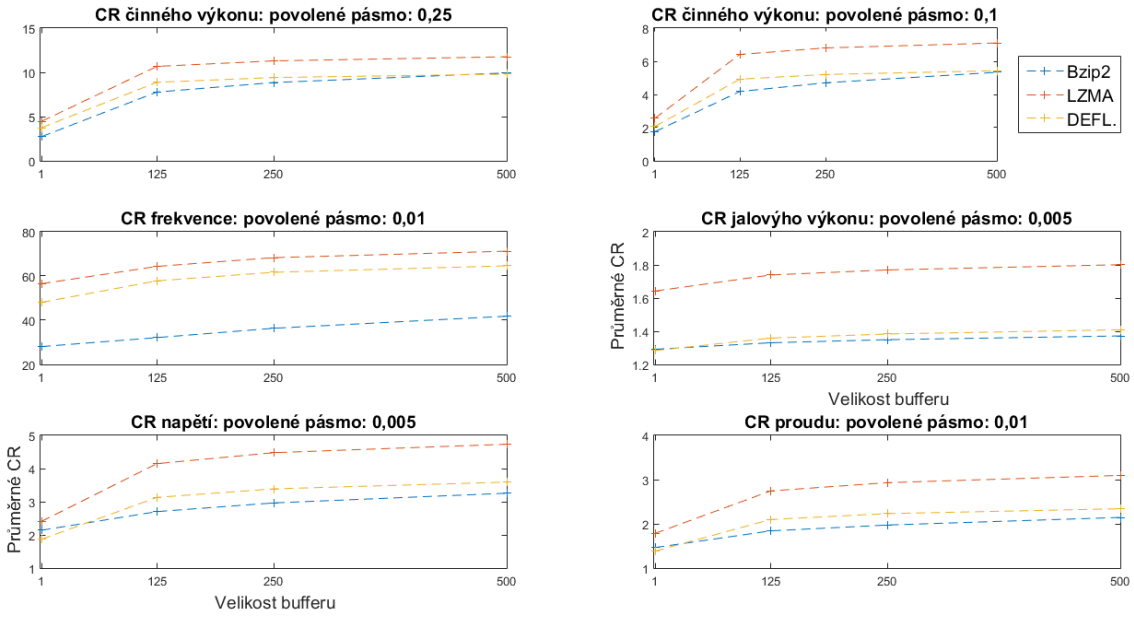
Otestuji vzorek 51 CEA souborů (jiné, než soubory z minulých testů). Naměřené veličiny z nich exportuji do binárních souborů o velikosti 1MB a následně zkomprimuji do bz2, 7z a zip formátu. K předběžnému zpracování dat využiji buffer předešlých hodnot tak, jak je popsaný v předcházející části (5.2). Zvolené velikosti bufferů jsou 1, 125, 250 a 500. Tyto buffery různých velikostí budu testovat na čtyřech třídách přesnosti (2,5; 1; 0,1; 0,05), nicméně maximální přijatelná změna velikosti veličiny bude pouze desetina třídy přesnosti (0,25; 0,1; 0,01; 0,005). Podobně jako v minulých testech, i zde zaznamenám stupeň komprese algoritmů Lzma, Bzip2

a Deflate. K určení časové náročnosti bufferů budu při ukládání každého čísla sledovat časovou náročnost metody procházející buffer a hledající vhodnou velikost k uložení. V mém C# programu je napsaná následovně:

```
private float bufferTP(Single nactenahodnota, Single pasmoTP)
{
    Single value;
    for(int i = buffer.Count - 1; i >= 0; i--)
    {
        value = buffer.ElementAt(i);
        if (value > nactenahodnota &&
        (nactenahodnota + (nactenahodnota*pasmoTP)) > value)
        {
            nactenahodnota = value;
            break;
        }else if(value < nactenahodnota &&
        (nactenahodnota - (nactenahodnota*pasmoTP)) < value)
        {
             nactenahodnota = value;
             break;
        }
    }
    if (aktualni_delka_bufferu < max_velikost_bufferu)
    {
        buffer.Enqueue(nactenahodnota);
        aktualni_delka_bufferu++;
    }
    else
    {
        buffer.Dequeue();
        buffer.Enqueue(nactenahodnota);
    }
    return nactenahodnota;
}
```

5.5 Výsledky testů

5.5.1 Stupeň komprese

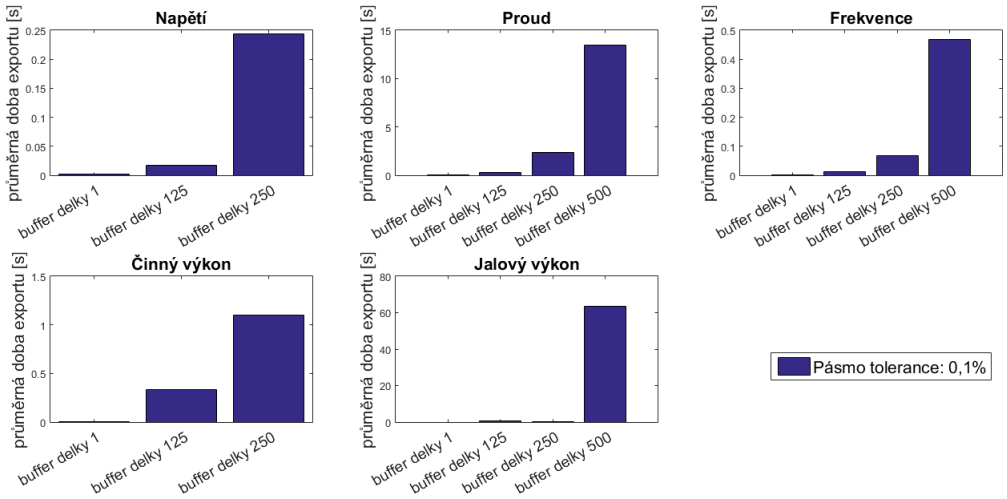


Obrázek 5.7: Změny v CR dle velikosti bufferu

Stupeň komprese se s použitím bufferů mnohonásobně zlepšil. Frekvence, jejíž CR se při ukládání v normální formě pohybovalo mezi 2 a 3 se při využití nejmenšího testovaného pásma tolerance (0,005%) pohybovala u Lzma, Bzip2 i Deflate v řádu desítek. Se zvětšením pásma tolerance se CR rapidně zvětšovalo až do velikosti v řádu tisíců. Jakožto veličina stále kolísající kolem 50Hz měla frekvence samozřejmě CR výrazně vyšší než ostatní veličiny. Přesto lze u všech veličin pozorovat některé společné rysy: Hlavní rozdíly v CR vznikaly zpravidla mezi buffery o velikosti 1 a 125. Mezi velikostmi 125, 250 a 500 se změny v CR rychle zmenšovaly i přes rostoucí časovou náročnost. (Algoritmus musel kvůli každému číslu procházet stále větší buffer.) Jako nejvhodnější velikost bufferu bych tedy volil čísla mezi 1 a 125. Dalším opakujícím se fenoménem byla jednoznačná nadřazenost algoritmu Lzma ve stupni komprese. Z grafů lze vidět, že nehledě na povolené pásmo a komprimovanou veličinu dosahoval Lzma stále nejlepších výsledků. Deflate se také výrazně zlepšil a končil zpravidla druhý. Bzip2 reagoval oproti předešlým dvěma algoritmům na tento druh předběžného zpracování výrazně nejméně. Tento rozdíl se s rostoucím CR zvětšoval a např. u frekvence s povoleném pásmem 0,1 a 0,25% dosahoval pouze zlomku stupně komprese ostatních algoritmů. Obě zmíněné vlastnosti lze pozorovat na grafech 5.7 zobrazujících CR různých veličin při pásmech tolerance.

5.5.2 Časová náročnost bufferů různé velikosti

Jak lze vidět na grafu časové náročnosti při povoleném pásmu 0,1% zde 5.8, časová náročnost rostla exponenciálně. Pokud do úvahy vezmeme i graf dosažených stupňů komprese (5.7), je jasné, že u bufferů délek 250 a 500 docházelo k častému procházení celého bufferu bez nalezení vhodné hodnoty. CR stagnuje na podobných hodnotách jako u bufferu velikosti 125 a časová náročnost exponenciálně roste. Proto jsem toho názoru, že velikosti bufferu 250 a 500 byly v tomto případě zbytečně velké.



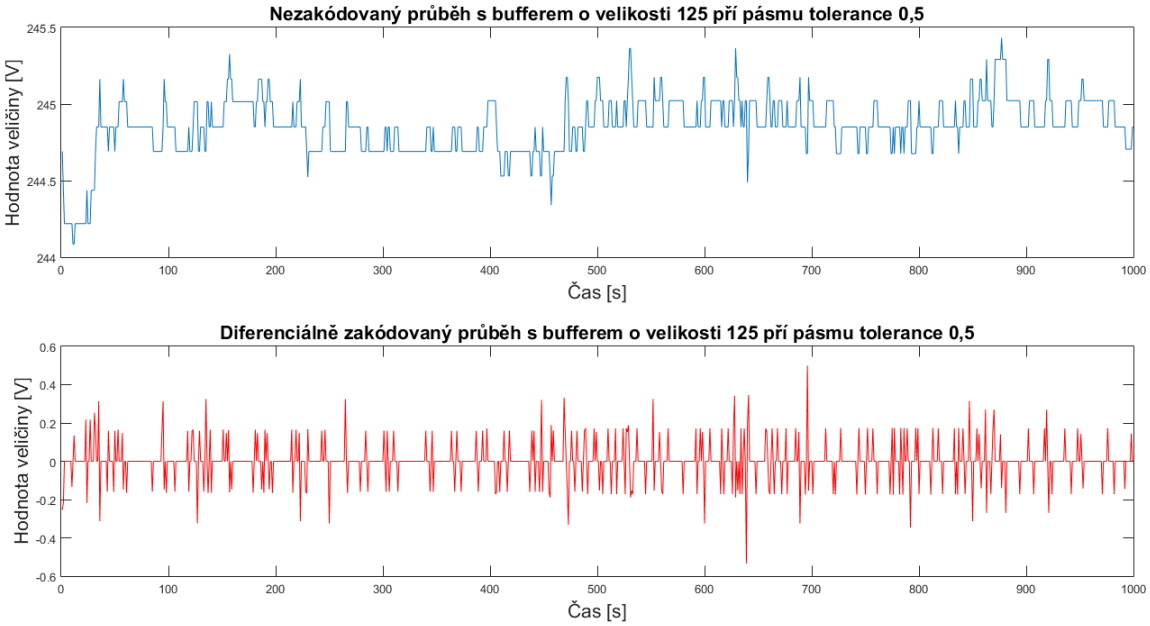
Obrázek 5.8: Průměrná časová náročnost bufferů různé velikosti při PT 0,1%

5.6 Buffer s diferenciálně zakódovanými hodnotami

Využití bufferu s předešlými hodnotami lze zkombinovat s diferenciálním zakódováním, které jsem testoval v kapitole 4.3. Jelikož se buffer prohledává od konce, namodelovaná měření mají často stejné hodnoty zřetězené za sebou. Při diferenciálním kódování, kdy se hodnota ukládá jako odchylka od minulého čísla, se všechny řetězce stejných hodnot uloží jako nuly (kromě prvního čísla).

Výhoda tohoto kódování tkví v častém výskytu nul, které se komprimačním algoritmům velmi dobře zpracovávají. Nevýhodou je výskyt kladných a záporných čísel, která se často střídají. Takový průběh může být sám o sobě horší ke komprimaci. Protože se neukládají přímé hodnoty, ale odchylky od minulých hodnot, průběh bude obsahovat více různých hodnot, než nezakódovaný průběh. Jak lze vidět na obrázku 5.9, střídání dvou různých hodnot v nezakódovaném průběhu se uloží jako tři rozdílné hodnoty: odchylka na novou hodnotu, nuly opakující se dokud hodnota zůstává stejná a následná odchylka na původní hodnotu, která má ale jinou velikost než první odchylka. (První a druhá odchylka mají stejnou velikost, ale rozdílná znaménka, přestože hodnota před první odchylkou a po druhé odchylce je v nezakódovaném průběhu stejná.)

Jelikož má delta kódování výše uvedené výhody i nevýhody, nejsem schopen dopředu určit, zda bude výhodné ho použít. Proto bude vhodné ho otestovat a provést analýzu při použití na reálných datech.



Obrázek 5.9: Průběhy ukládaného napětí při pásmu tolerance 0,5 s využitím bufferu předešlých hodnot

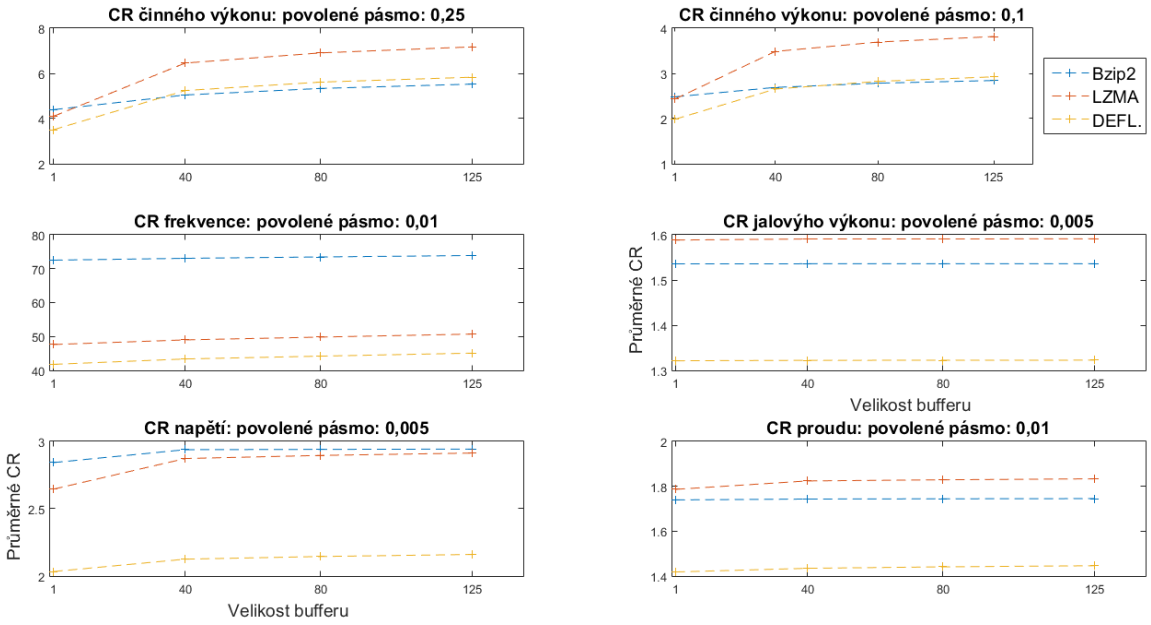
5.7 Testování bufferu s diferenciálním kódováním

Cílem těchto testů je otestovat techniku popsanou v minulé sekci, tedy buffer s delta kódovanými hodnotami. Testována bude skupina stejných 52 CEA souborů jako u testů bufferu s nezakódovanými čísly (viz 5.4), aby nebyly rozdíly mezi těmito testy způsobeny jinými daty.

Hlavní otázkou přirozeně zůstává, zda se zakódovaná data budou lépe komprimovat. Jedním ze závěrů předchozích testů bylo konstatování, že stupeň komprese se zvětšoval hlavně mezi buffery o velikostech 1 a 125. Delším bufferům se CR zvyšovalo pouze minimálně, časová náročnost přitom rostla exponenciálně a jejich využití nemělo smysl. Jelikož vím, že ideální délka bufferu se nachází mezi hodnotami 1 a 125, budou tyto testy prováděny na bufferech 1, 40, 80 a 125, abych mohl vhodnou délku zvolit podle přesnějších dat. Otestují se opět povolená pásma 0,25; 0,1;0,01 a 0,05.

5.8 Výsledky testů

CR se při zakódování téměř universálně zmenšilo. K ilustraci poslouží obr. 5.10, kde jsou zobrazeny stejné veličiny se stejným povoleným pásmem jako u minulého testu. Buffery o velikosti 1 a 125 dosahují podobného nebo menšího CR. Na rozdíl od nezakódovaného bufferu, kde byl LZMA vždy nejlepší, v tomto testu dosahoval Bzip2 lepšího CR než LZMA u frekvence a u napětí. Tento efekt bylo možné pozorovat již u prvních testů, kde se buffer nevyužíval.



Obrázek 5.10: Změny v CR dle velikosti diferenciálně kódovaného bufferu

5.9 Shrnutí

- Použití bufferu se ukázalo jako smysluplné. Zvětšování velikosti bufferu přes 100 již smysl nemělo. Po analýze jsem pro mé použití zvolil za nejvhodnější buffer o velikosti 80.
- Při normálním kódování se u všech veličin nejvíce osvědčil algoritmus Lzma. Proto bych jako nejvhodnější parametry této kompresní techniky při použití na mé archivy použil velikost bufferu 80 a následnou kompresi pomocí Lzma.
- Buffer bez nutnosti jakékoliv změny v provedení pružně reaguje na povolené pásmo tolerance a jeho efektivita se podle něj také mění.

- Velikost RMSE a MSE se lineárně zvyšovala se zvětšujícím se povoleným pásmem.
- Spojení bufferu a delta kódování, které jsem zkoumal v minulé sérii testů, se neosvědčilo. Důvodem je zvětšení počtu různých čísel vyskytujících se v průběhu při delta kódování.
- Původně jsem zamýšlel pokusit se využít i plovoucí či pevnou řádovou čárku, ale ani jeden způsob se mi nepodařil uspokojivě implementovat, protože se mi nezdařilo vyřešit problém počtu desetinných míst. Vzhledem k rozmanitosti ukládaných dat jsem tedy usoudil, že využití těchto způsobů reprezentace dat v bufferu nebude výhodné.

6 Závěr

Výsledky této práce se dají rozdělit do několika kategorií.

První kategorií je otestování známých kompresních algoritmů při práci s různými kódováními měření veličin elektrické energie. Zpráva obsahuje záznamy o testování diferenciálního kódování, pevné řádové čárky a jejich kombinace. Dále jsou zde uvedeny výsledky kompresních algoritmů Lzma, Bzip2 a Deflate, byly popsány jejich silné a slabé stránky. Všechny zmíněné techniky byly ozkoušeny na souborech různé velikosti. Jak už bylo mnohokrát v práci zmíněno, výsledky nejsou jednostranné a nejvhodnější způsob komprese záleží na tom, co uživatel od komprese očekává. V této bakalářské práci, ve které se psaly a testovaly kompresní programy v jazyce C# pro využití na počítačích, měl stupeň komprese větší prioritu než časová náročnost. Proto jsem v první sérii testů za nejvhodnější způsob komprese označil kombinace diferenciálního kódování a pevné řádové čárky použité v souborech velkých 1MB a následná komprese algoritmem Lzma. Takovýto postup dosahoval velmi dobrých výsledků při práci se všemi archivovanými veličinami.

Do druhé kategorie patří seznámení a otestování nového způsobu ztrátové komprese, jenž je založen na úpravě dat podle třídy přesnosti. K tomu se využíval buffer minulých hodnot. Tato technika ukládání se rovněž osvědčila. Její výhodou je, že s větší dovolenou třídou přesnosti a s velikostí bufferu se výsledky budou zlepšovat. Může být tedy na uživateli, aby se v závislosti na výpočetním výkonu, druhu archivovaných dat a potřebné přesnosti měření rozhodl pro zvolené parametry bufferu. Tato oblast skýtá ještě značný prostor pro zlepšení, aby buffer fungoval efektivněji, např. lepší způsob rozhodování, která data do bufferu budeme ukládat. Další oblast, kde je možné na tuto práci navázat, je zkombinování bufferu a dalších kompresních technik. Jeden ze způsobu zlepšení komprese, kterým jsem se zabýval, byla kombinace bufferu a diferenciálního kódování. Tento postup ale sám o sobě nebyl úspěšný. Je tedy možné řešení upravit a využití bufferu spolu s dalšími technikami zdokonalit.

Hlavní smysl této práce vidím především v prozkoumávání možností komprese dle třídy přesnosti měřících přístrojů, která zůstává neprobádaným teritoriem ztrátové komprese. Doufám, že i tato práce se malým dílem zaslouží o změnu statu quo, protože v této oblasti je jistě ukryt velký potenciál pro využití i v průmyslovém prostředí.

Seznam použité literatury

[1] P. Tišnovský. *Fixed point arithmetic*. May 2006. URL: https://www.root.cz/serialy/fixed-point-arithmetic/.

[2] Stephanie. *RMSE: Root Mean Square Error*. Oct. 2016. URL: https://www.statisticshowto.datasciencecentral.com/rmse/.

[3] JJ. *MAE and RMSE — Which Metric is Better? Mean Absolute Error versus Root Mean Squared Error*. Mar. 2016. URL: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

[4] J. Ning, J. Wang, W. Gao, and C. Liu. "A Wavelet-Based Data CompressionTechnique for Smart Grid". In: *212IEEE TRANSACTIONS ON SMART GRID* 2.1 (Mar. 2011).

[5] J. Khan, S. Bhuiyan, G. Murphy, and M. Arline. "Embedded zerotree wavelet based data compression for smart grid". In: *2013 IEEE Industry Applications Society Annual Meeting* (Oct. 2013).

[6] C. S. Lai. "Compression of power system signals with wavelets". In: *Proceedings of the 2014 International Conference on Wavelet Analysis and Pattern Recognition* (July 2014).

[7] M. P. Tcheou, L. Lovisolo, MoisesV.Ribeiro, E. A. B. da Silva, M. A. M. Rodrigues, J. M. T. Romano, and P. S. R. Diniz. "The Compression of Electric Signal Waveforms forSmart Grids: State of the Art and Future Trends". In: *IEEE Transactions on Smart Grid* 5.1 (Jan. 2014).

[8] J. C. S. de Souza, atiana Mariano Lessa Assis, and B. C. Pal. "Data Compression in Smart Distribution Systems via Singular Value Decomposition". In: *IEEE TRANSACTIONS ON SMART GRID* 8.1 (Jan. 2017).

[9] R. E. Dapper, A. A. Susin, S. Bampi, and C. D. P. Crovato. "High compression ratio algorithm for power quality signals". In: *2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)* (June 2015).

[10] M. Ringwelski, C. Renner, A. Reinhardt, A. Weigel, and V. Turau. "The Hitchhiker's guide to choosing the compression algorithm for your smart meter data". In: *2012 IEEE International Energy Conference and Exhibition (ENERGYCON)* (Sept. 2012).

[11] R. Klump, P. Agarwal, J. E. Tate, and H. Khurana. “Lossless Compression of Synchronized Phasor Measurements”. In: *IEEE PES General Meeting* (July 2010).

[12] Z. B. Tariq, N. Arshad, and M. Nabeel. “Enhanced LZMA and BZIP2 for Improved Energy Data Compression”. In: (May 2015).

[13] A. Moffat. “Implementing the PPM data compression scheme”. In: *IEEE TRANSACTIONS ON COMMUNICATIONS* 38.11 (Nov. 1990).

[14] J. Seward. *bzip2*. URL: http://www.bzip.org.

[15] J. Kraus, T. Tobiška, and V. Bubla. “Loosles encodings and compression algorithms applied on power quality datasets”. In: (June 2009).

[16] J. Murray and W. vanRyper. *Encyclopedia of Graphics File Formats, 2nd Edition*. O’Reilly Media, 1996. ISBN: 1-56592-161-5.

[17] M. Burrows and D. Wheeler. “A Block-sorting LosslessData Compression Algorithm”. DEC Systems Research Center, May 1994.

[18] B. Ryabko. “Data compression by means of a ’book stack’”. In: *Problems of Information Transmission* (1980).

[19] D. A. Huffman. “A Method for the Construction of Minimum-Redundancy Codes”. In: *Proceedings of the IEEE* (1952).

[20] I. Pavlov. *lzma_sdk*. 2018. Chap. LZMA SDK 18.05.

[21] I. Pavlov. *LZMA spec?* URL: https://sourceforge.net/p/scoremanager/discussion/457976/thread/c262da00/.

[22] J. Kraus, P. Stepan, and L. Kukacka. “Optimal data compression techniques for smart grid and power quality trend data”. In: (2012).

[23] J.-l. Gailly. *GNU Gzip*. URL: https://www.gnu.org/software/gzip/manual/gzip.html.

[24] J. Ziv and A. Lempel. “A universal algorithm for sequential data compression”. In: *IEEE Transactions on Information Theory* (May 1977).

[25] A. Yazdanpanah and M. R. Hashemi. “A simple lossless preprocessing algorithm for hardware implementation of DEFLATE data compression”. In: *2011 19th Iranian Conference on Electrical Engineering* (July 2011).

[26] Microsoft. *GZipStream Class*. URL: https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.gzipstream?view=netframework-4.7.2.

[27] Microsoft. *DeflateStream Class*. URL: https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.deflatestream?view=netframework-4.7.2.

[28] J. Moravec. “Zpracování a vizualizace dat analyzátorů v OS Android”. 2017. Chap. CEA soubor.

[29] J. Kraus and V. Bubla. “Optimal Methods for Data Storage in Performance Measuring and Monitoring Devices”. In: (Jan. 2008).